Introduction générale et problématique

L'industrie a connu ces dernières années des progrès technologiques inégalables à un rythme très rapide, ce qui a pour but d'accélérer et d'améliorer la production, mais cette remarquable avancée a engendré des problématiques, le personnel qui travaille dans cette industrie a parfois du mal à s'adapter ou d'avoir une maitrise excellente de ces nouvelles technologies pour faire des vérifications ou des maintenances sur ces dernières.

Malgré plusieurs formations trop souvent théoriques, l'expertise de ces travailleurs demeure insuffisante et surtout pour les jeunes recrues (Jeunes diplômés) qui n'ont pas encore l'expérience du terrain et ont besoin de l'assistance de personnes plus expérimentées.

Les travailleurs avec une expérience ou des qualifications pertinentes peuvent se retrouver sur des sites très éloignés géographiquement de ceux des novices, et ainsi la E-Maintenance rentre en jeu, grâce à une connectivité (Internet par exemple) ils peuvent rentrer en communication pour guider un autre employé moins bien expérimenté pour la maintenance d'une pièce, d'un engin par exemple.

Une fois cette connectivité établie, un deuxième problème se pose, les experts ne sont pas forcément de bons pédagogues pour bien expliquer au novice ce qu'il faut changer ou enlever ou ajouter à une pièce précise d'une machine, on a donc besoin de leur apporter une boite à outils « couteau suisse » qui regroupe le matériel nécessaire aidant à guider le novice. Ainsi si le matériel est représenté en 2D, il peut porter à confusion, et être positionné à distance par l'expert sur le mauvais endroit de la pièce machine qui se trouve chez le novice et sur laquelle doit s'appliquer la maintenance.

Nous allons mettre en œuvre un système de E-maintenance pour le CDTA (Centre de Développement des Technologies Avancées), en commençant par faire une étude sur ce qu'est la E-maintenance et la réalité augmentée dans le premier chapitre, et voir quelques notions sur la transmission vidéo sur le réseau internet dans le deuxième chapitre.

Ensuite dans le troisième chapitre, nous proposerons des approches différentes pour réaliser ce système, et détaillerons sa conception. Nous allons voir aussi comment utiliser la réalité augmentée pour avoir des objets d'interaction en 3D (Pinces, marteau etc.), et comment assurer une bonne connectivité pour partager la scène du novice vers l'expert, afin que ce dernier puisse guider et voir les changements en temps réel se déroulant chez le novice.

Introduction

Le quatrième chapitre décrit notre application suite à la conception que nous avons effectuée lors du chapitre 3.

Enfin, nous clôturons notre mémoire par une conclusion générale et des perspectives.

Figure 1 : E-Maintenance & Smart Glass [1]

Chapitre 1

1. E-maintenance et réalité augmentée

1.1 Introduction à la e-maintenance :

La Maintenance a pour objectif d'assurer le bon fonctionnement d'un service et cela par des vérifications, des contrôles, des révisions, voir même des réparations et des dépannages si nécessaires.

La maintenance peut être appliquée sur des équipements logiciels ou matériels, la maintenance du matériel est assez délicate car en plus de la maitrise de l'architecture et de la connaissance du fonctionnement de l'engin sur lequel s'effectue la maintenance, l'ingénieur doit aussi avoir une bonne visibilité sur l'engin afin de pouvoir diagnostiquer le problème et de savoir à quel endroit précisément faire des tests ou des réparations ou autre action de maintenance.

Parfois sur des engins de dernières technologies ou quand des ingénieurs ne sont pas assez qualifiés sur des types d'engins, ils ont besoin d'une assistance complète de la part d'un expert pour réussir leurs missions, et ce même si des millions de kilomètres les séparent.

Pour ce faire, l'expert doit avoir une visibilité parfaite de cet objet, et donc peut voir ce que le novice voit, et a la possibilité d'interagir avec le novice afin de le guider pour effectuer la maintenance à distance, c'est ce qui caractérise la E-Maintenance.

Littéralement parlant, la E-maintenance possède plusieurs définitions, mais il y a un critère commun entre les différentes définitions, c'est l'utilisation des technologies du web (internet) et la technologie de l'infotronique pour les tâches de maintenance ainsi que la nécessité d'avoir une infrastructure fiable et une architecture fiable pour l'implémentation de l'e-maintenance [2].

La définition la plus généralisée selon [3] : « La e-maintenance est un concept de gestion de la maintenance selon laquelle les actifs sont suivis et gérés sur internet. Elle introduit un niveau de transparence sans précédent et qui est efficace dans toutes les industries. »

Le préfixe "e" que l'on prononce souvent en anglais "i" du terme e-maintenance, est l'abréviation d'"***électronique***" mais peut aussi évoquer l'expression "***en ligne***", utilisée avec le terme qui suit, qui peut être e-maintenance, e-learning etc.

Lors d'une maintenance à distance, pour que l'expert puisse guider le novice, il faut des outils d'interaction capable de se positionner à un endroit précis sur l'image partagée afin que le novice sache sur quel endroit de l'engin, où la maintenance se fait, il faut intervenir. Il est donc

nécessaire d'utiliser des objets 3D virtuels, pour cela, nous utiliserons la réalité augmentée que nous détaillerons par la suite dans ce chapitre.

1.2 Plateformes de maintenance développées

1.2.1 Plateforme PROTEUS :

C'est un projet européen, la plateforme PROTEUS est une plateforme de e-maintenance qui propose des systèmes d'aide à la maintenance à travers des services web, elle est développée à l'aide des technologies d'Internet et permet notamment la collaboration des acteurs, le couplage avec les outils classiques de gestion d'entreprise de type ERP (Enterprise Resource Planning), l'aide au diagnostic, l'accessibilité à diverses ressources (bases de données), etc.

1.2.2 Plateforme TELMA :

La plateforme pédagogique et de recherche TELMA [4] (TELé-Maintenance) est développée par le CRAN (Centre de Recherche en Automatique de Nancy), elle a pour objectif de supporter les enseignements de la maintenance et d'illustrer l'apport des nouvelles technologies de l'information et de la communication sur les processus et organisations de maintenance.

Elle est utilisable en local et à distance et est capable de générer un ensemble de défaillances qui peuvent nuire au bon fonctionnement.

1.2.3 Plateforme TATEM :

Le projet TATEM [5] (Technologies And Techniques for nEw Maintenance concepts), financé par l'Union Européenne vise à évaluer les technologies capables d'augmenter l'exploitabilité des avions par un transfert de la maintenance non programmée vers une maintenance programmée et par une amélioration du rendement et de l'efficacité du processus de maintenance.

Il s'agit d'intégrer un système de surveillance et de gestion de l'état de santé des avions mis en place dans le cadre d'un plan de gestion intégré avait le potentiel de réduire considérablement les coûts d'exploitation liés à la maintenance aéronautique. Dans le but d'avoir des effets positifs sur la sécurité des voyageurs et les conditions de travail des mécaniciens aéronautiques, et profiter aux compagnies tout en stimulant fortement l'industrie aéronautique européenne.

1.2.4 Plateforme NEMOSYS :

NEMOSYS [1] (naval E-Maintenance Oriented SYStem) est une plateforme informatique pour la e-maintenance sur des systèmes navals de défense développée à la DCN (Direction des Constructions Navales) par le centre américain IMS (Intelligent Maintenance System).

Le système de maintenance intelligent est un système embarqué, collaboratif et distribué dans lequel sont effectués des échanges d'informations en temps réels et à distance entre les navires et la terre.

1.2.5 Plateforme GAMA-Frame :

GAMA-FRAME [6] (Global Asset MAintenance FRAME work) est une plateforme de e-maintenance qui intègre un module de diagnostic par RAPC qui est une approche de résolution de problèmes et d'apprentissage dans le but de faire de la maintenance corrective et plus particulièrement au diagnostic de pannes des équipements industriels,

1.2.6 Service e-maintenance de Canon :

Canon e-Maintenance [7] est une solution qui automatise de nombreuses tâches fastidieuses nécessaires à la gestion des périphériques réseau multifonctions.

Canon propose ce service dans le but de réduire les tâches administratives grâce aux relevés automatisés, et de prendre le contrôle des appareils des utilisateurs et leur faire gagner du temps grâce au système RDS de diagnostic à distance (Remote Diagnostic System), si une défaillance se produit ou qu'un périphérique ne fonctionne plus correctement par exemple, le fournisseur de service est immédiatement et automatiquement averti par e-mail.

1.3 Potentiel des technologies émergentes dans l'aide à la maintenance

1.3.1 La réalité virtuelle :

La réalité virtuelle (RV) est une technologie immersive qui permet de simuler un monde virtuel pour faire vivre à l'utilisateur une expérience d'immersion dans cet environnement artificiel en utilisant des outils informatiques, l'utilisateur a la possibilité d'interagir avec les différents objets virtuels qui composent cet univers irréel.

La RV peut être utile dans différents domaines, les domaines le plus envahis par la RV sont les domaines de divertissement, elle est utilisée pour rendre le plaisir plus intense aux internautes ou aux gamers dans les jeux vidéo, désormais grâce à la réalité virtuelle, le joueur peut participer au jeu dont il est le personnage principal et le vivre en temps réel comme s'il était dedans.

D'autres domaines où l'on peut retrouver la RV comme en architecture pour aider les architectes à être plus précis dans leur travail afin de satisfaire leurs clients, mais aussi la science, la communication, la mode, le sport, l'aviation, la médecine ou encore tout secteur d'enseignement pour former les élèves dans la pratique, il leur est maintenant possible d'apprendre sans avoir peur de l'échec.

Ses principaux outils sont les casques de réalité virtuelle comme l'Oculus Rift.

Figure 2 : Oculus rhift

1.3.2 La réalité augmentée

1.3.2.1 Contexte de la maintenance distante :

La maintenance à distance est une forme évoluée de la maintenance, elle permet de rendre disponibles des ressources à distance et de pouvoir les contrôler, ces ressources-là sont accessibles grâce à un réseau de communication comme le téléphone ou Internet dans le but de diagnostiquer, gérer, résoudre des problèmes, ou alors d'administrer un système.

La maintenance distante concerne plusieurs domaines d'applications industriels comme la navigation aérienne, la navigation navale, les appareils et périphériques aériens, les plateformes pétrolières, le Bâtiment, la sécurité civile et publique, la Défense etc.

En général, elle se fait par le biais du réseau Internet, lors d'une coopération technique à distance entre un expert situé dans un centre technique et un technicien présent sur le terrain par exemple, et ce dernier sera chargé d'intervenir sur une machine.

Quand la maintenance se fait par vidéo, on parle de maintenance assistée par vidéo : VAM (video-assisted maintenance), l'intervenant sur le terrain doit communiquer par une caméra seulement, et l'expert visualise instantanément le flux vidéo transmis. L'expert et le technicien doivent partager en temps réel les mêmes scènes.

D'autres fonctionnalités peuvent s'ajouter au VAM, comme un chat vocal, et/ou un chat écrit afin d'obtenir une coopération technique efficace, et renforcée par la capacité qu'a l'expert de superposer des objets de réalité augmentée sur les images reçues, toutes ces données-là sont partagées à travers le réseau Internet, d'où le terme **e-maintenance**.

1.3.2.2 Définition de la réalité augmentée :

La réalité augmentée est à distinguer de la réalité virtuelle qui s'agit d'une technologie de simulation qui, contrairement à la réalité augmentée, s'appuie majoritairement sur le virtuel.

La réalité augmentée (RA) désigne une réalité telle que nous l'apercevons mais en ajoutant un plus, une augmentation qui représente un objet virtuel, cet objet sera positionné dans le monde réel grâce à un repère 3D (x,y,z) qu'on appelle marqueur, et qui peut représenter soit une image sur une feuille ou un objet réel 3D (Une bouteille, une trousse etc..) mais ce dernier n'est pas forcément obligatoire, on peut utiliser un Markerless, où le marqueur n'existe pas, et l'objet est créé par rapport à la terre par exemple (Ground Plane).

La RA représente en point de vue informatique, un traitement d'image avancé qui peut d'abord détecter le marqueur, ensuite lui associe un objet virtuel 3D et le positionner, ces traitements se font en temps réel et sont un peu couteux en matière de ressource, ils peuvent se faire grâce à des caméras et un processeur (PC, Tablette, Mobile, Lunette Hololens Les visiocasques.).

Ronald Azuma, défini la réalité augmentée comme : *« un système de réalité augmentée complète le monde réel avec des objets virtuels (générés par ordinateur) de telle sorte qu'ils semblent coexister dans le même espace que le monde réel »* [8].

Il existe différents domaines où la RA s'impose comme un outil indispensable d'assistance et d'aide à la décision, dans le domaine médical, elle permet de faire apparaître les organes en superposition sur le corps humain et de créer des simulations interactives d'interventions chirurgicales, comme elle est utilisée aussi dans l'industrie, le tourisme, le divertissement et même dans le domaine militaire.

Voici quelques exemples d'applications réalisées dans un environnement de RA :

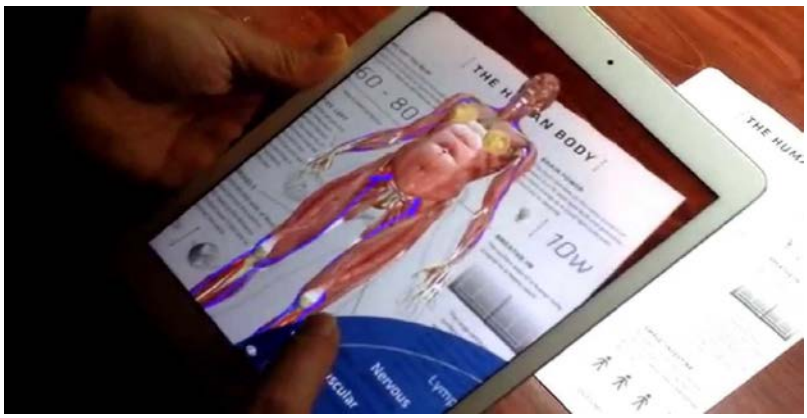


Figure 3 : Anatomy 4D – Application médicale

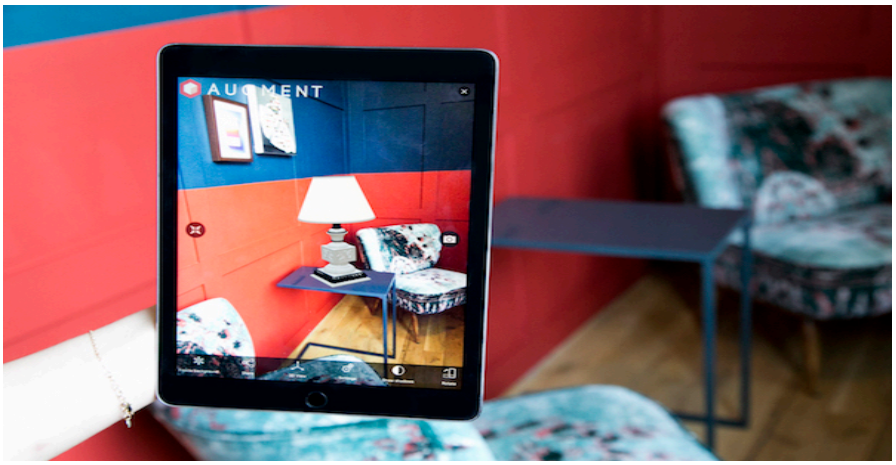


Figure 4 : Augment – Application pour décoration d'intérieur de maison

1.3.2.3 Principe de fonctionnement de la réalité augmentée :

Pour faire de la réalité augmentée il faut : une caméra de capture, un système d'exploitation, un écran pour diffuser l'information et un SDK (Software Developpement Kit) appellé aussi moteur à RA, son rôle et de faire tous les calculs pour permettre à la machine de comprendre ce que voit la caméra.

- En premier lieu, il faut mettre un marqueur dans la scène à filmer, un marqueur est une image qui devra être reconnue par notre SDK.
- Ensuite la caméra filme la scène réelle (sous formes d'images).
- Chaque image capturée passe par deux étapes cruciales, le recalage et le suivi de mouvement [9] (tracking en anglais) où le marqueur cible est recherché dans la scène réelle, une fois détecté la position et l'orientation de la caméra est calculée et un système de coordonnées de trois axes (x,y,z) est créé dans le monde réel, pour suivre les mouvements de la caméra des algorithmes de suivi de points d'intérêts sont utilisés afin de permettre d'estimer la pose de la caméra à chaque rafraichissement, cette étape est important afin de savoir où positionner l'objet 3D de façon cohérente avec le monde réel dans les prochaines étapes.
- Une fois que le tracking réalisé, les objets virtuels sont ajoutés en cohérence avec le monde réel et le rendu est mixé sur l'écran de l'appareil d'affichage.

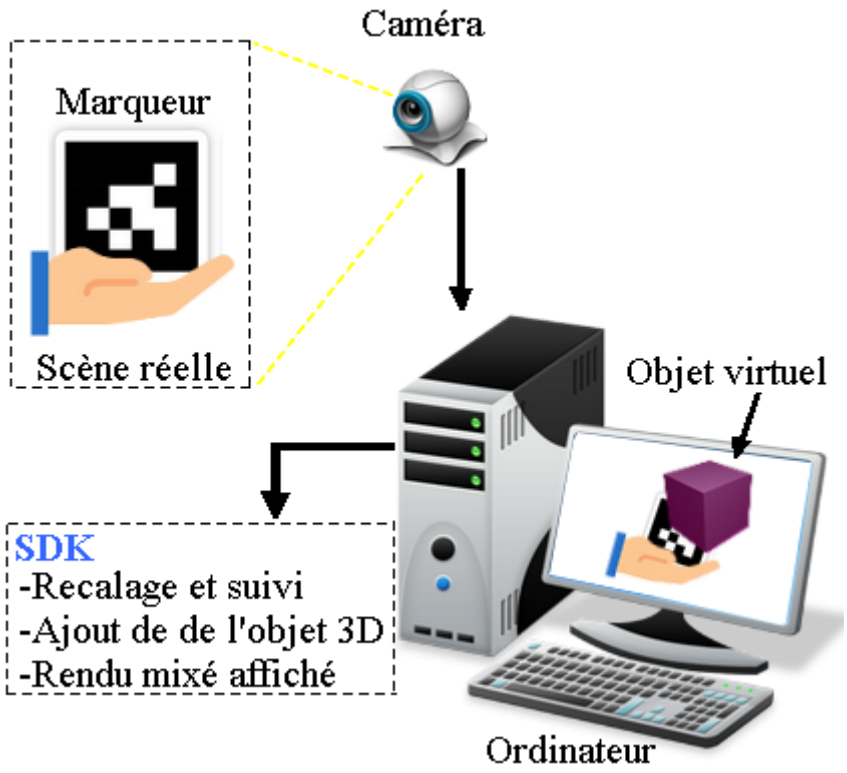


Figure 5 : Fonctionnement de la réalité augmentée

1.3.2.4 Les méthodes de suivis (tracking) dans la RA

Il existe de nombreuses méthodes de tracking en réalité augmentée :

- **Méthodes utilisant des capteurs :**

Il s'agit d'utiliser des capteurs pour déterminer la position des objets dans l'environnement, il existe plusieurs types de capteurs : magnétiques, acoustiques, inertiels, GPS …

Ces techniques ont l'avantage d'être robustes aux mouvements rapide de la caméra, mais ils ont aussi l'inconvénient d'être sensibles aux perturbations de l'environnement [10].

- **Méthodes utilisant des marqueurs :**

Ces méthodes utilisent la vision par ordinateur en général basée sur la détection des positions par des caméras, elles consistent à utiliser un marqueur utilisant des formes pour lui permettre d'être repéré facilement dans une scène filmée et déduire en suite la position de la caméra et faire correspondance entre un point 3d de la scène et son image 2d, en général ils utilisent des formes géométriques pour représenter le centre de l'espace 3D.

Les méthodes avec marqueur sont connues pour leur rapidité d'exécution de l'algorithme de détection et la stabilité de la détection, mais la présence du marqueur dans l'environnement pose problème, ils sont sensibles à la luminosité et restreints à leur zone de visibilité, ils peuvent devenir indétectables lorsque la caméra s'éloigne.

- **Méthodes utilisant des marqueurs naturels :**

Ces méthodes de suivi sont basées sur des images naturelles qui sont déjà présente dans l'environnement réel, il n'est pas nécessaire de préparer l'environnement en y plaçant des marqueurs pour permettre au système de RA de se repérer dans l'environnement.

On retrouve plusieurs types de méthode qui utilisent ces marqueurs [12] :

- Méthodes utilisant les contours
- Méthodes utilisant les coins
- Méthodes utilisant des points d'intérêt
- Méthodes utilisant des motifs texturés

Ces méthodes-là sont aussi appelés méthodes sans marqueur car elles ne dépendent pas de ce dernier, mais présentent quand même un inconvénient car elles utilisent des algorithmes de vision par ordinateur et la plupart consomment beaucoup en ressources de calcul.

- **Méthodes de suivi hybrides :**

Etant donné que les méthodes de suivis possèdent toutes des intérêts et des limites, il est possible de développer des méthodes hybrides [11] combinant plusieurs approches.

L'idée est de faire, selon les nécessités, combiner différentes technologies et méthodes de suivis et pourquoi pas aussi avec un ou plusieurs capteurs.

1.3.3 Plateforme de maintenance basée sur la réalité mixte « ARIMA » du CDTA :

ARIMA [13] (AugmentedReality and Image processing in e-Maintenance Application) est un système se base sur le concept de la RA pour aider les techniciens à accroître leur efficacité dans des activités de e-maintenance.

Il s'agit de mettre en place un environnement de travail pour l'aide à la maintenance pour permettre à des techniciens de manipuler des documents électroniques avec des outils réels.

La procédure de maintenance se passe comme suit :

- L'intervenant doit d'abord se connecter à l'application pour accéder à son planning, et sur le lieu de l'intervention, il récupère les informations concernant la machine, l'historique des interventions, les derniers intervenants, etc… et à partir des informations acquises
- Le technicien lance la procédure de maintenance qui correspond à la machine.
- Le scénario de maintenance est transmis sous forme d'augmentations au niveau des dispositifs de RA chez le technicien.

- Quand l'intervention est effectuée, le technicien envoie au superviseur et remplit un rapport de maintenance pour spécifier le type de l'intervention et la panne.
- Après cela le superviseur met à jour le planning d'intervention en intégrant le nouveau rapport.
- A la fin de la maintenance, l'historique est sauvegardé.

Dans certains cas exigeant, le technicien collabore à distance avec un expert qui accède à l'historique de l'intervention et guide le technicien dans sa maintenance via des indications graphiques ou sous formes de textes, ou alors par un chat vocal.

1.3.4 Conclusion :

Le développement technologique a permis de moderniser et d'augmenter les productivités de l'industrie mais il a apporté avec lui son lot de désagrément, notamment pour la maintenance de ces nouveaux engins technologiques.

Néanmoins, en associant plusieurs technologies, nous pouvons proposer des solutions concrètes ainsi en combinant la fonction de maintenance à des outils de réalité augmentée, qui offrent plusieurs alternatives (Avec marqueur, sans marqueur etc.) la solution aux problèmes de maintenance est bien claire : C'est la E-Maintenance mais comment la mettre en œuvre choisir les bons outils, les bons protocoles et architecture pour avoir un système optimal et fonctionnel en temps réel ?

Chapitre 2

2. Communication vidéo et réseau

2.1 Introduction :

La communication ne se fait plus seulement sur des supports en papier aujourd'hui, elle se développe à très grande vitesse sur Internet grâce à la diversité des équipements électroniques qui permettent d'enregistrer, partager, diffuser, et de lire des contenus multimédias (musique, image, vidéo…) et surtout depuis que l'analogique a cédé sa place au numérique.

Le transfert du flux vidéo sur Internet diffère des techniques de transmissions classiques telles que la radio et la télévision, bien que les concepts employés soient similaires.

En effet, les données multimédias, notamment les vidéos, sont volumineuses et imposent des contraintes sur le réseau qui ne correspondent pas à certains protocoles réseaux, infligeant des problèmes de bande passante, de débit, et bien d'autres problèmes de de représentation de la vidéo.

Nous tenterons, lors de ce 2^{e} chapitre de résumer certaines caractéristiques sur la transmission vidéo sur le réseau ainsi que des éléments de solutions pour répondre à ces problèmes.

2.2 Communication vidéo :

2.2.1 Techniques de diffusion vidéo :

a) Le téléchargement :

Ce moyen consiste à enregistrer la vidéo complètement chez l'utilisateur, l'avantage du téléchargement est que l'utilisateur peut revoir le contenu de sa vidéo autant de fois qu'il le désire une fois le contenu téléchargé, avec une qualité déterminée lors de l'encodage.

Une taille importante est permise dans ce cas-là face à un temps de téléchargement plus long, mais elle doit être raisonnable afin que les utilisateurs ayant un débit bas puissent la télécharger.

b) La diffusion en temps réel :

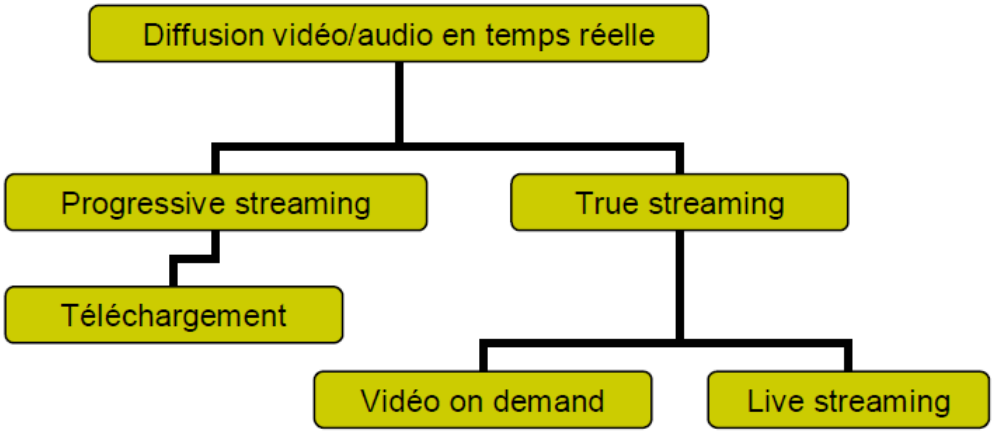


Figure 6 : Schéma diffusion vidéo/audio en temps réel [14]

- **Le streaming progressif :** Ou téléchargement progressif, le flux vidéo est enregistré au fur et à mesure de sa réception dans une mémoire tampon (buffer) chez l'utilisateur, mais il n'aura aucun contrôle sur le flux, c'est-à-dire : avancer, rembobiner le film…

Son avantage, c'est qu'on peut diffuser le stream à partir d'un serveur web avec le protocole HTTP sans fonctionnements supplémentaire.

- **Le true streaming :** Ou la vraie diffusion en temps réel, qui permet soit des diffusions « on demand », où le flux est diffusé sur demande, contrôlé par l'utilisateur.
  Ou alors le live streaming, qui se fait en direct, c'est-à-dire que la diffusion se fait en même temps que l'enregistrement du flux.

2.2.2 Compression et décompression vidéo :

L'opération du codage consiste à compresser la donnée au maximum de façon à pouvoir utiliser au mieux la capacité du canal de transmission utilisé.

L'étape de compression se fait en plusieurs étapes :

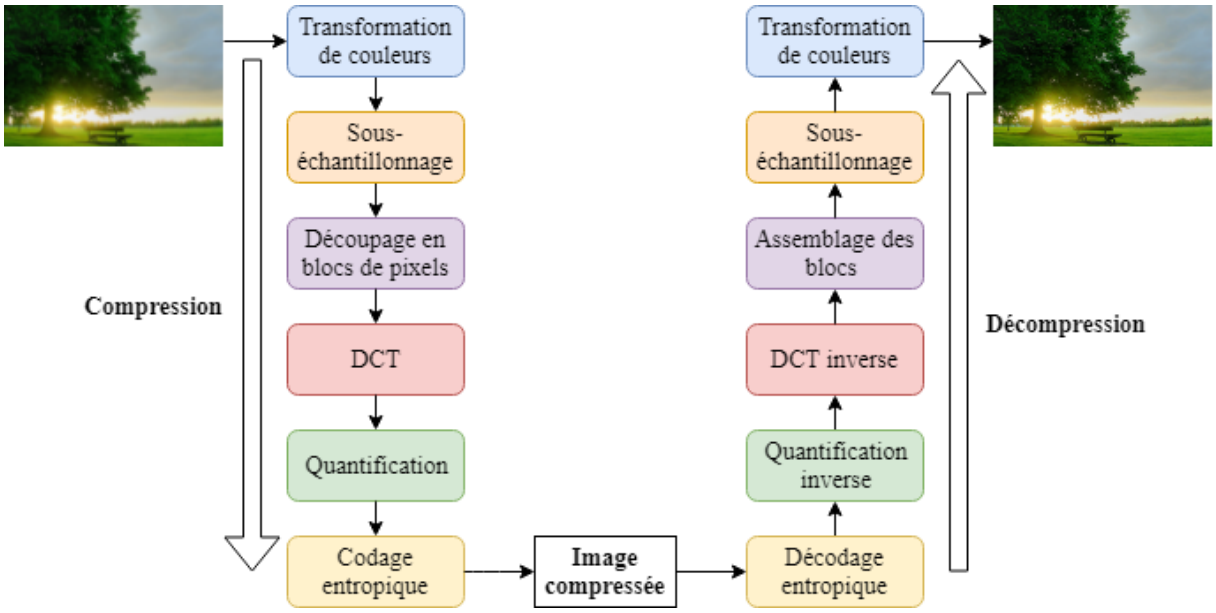


Figure 7 : Schéma des différentes étapes de la compression (et décompression)

1) **Transformation de couleur** : Passage du modèle initial des couleurs de l'image en format RGB (RVB en français) au modèle de type chrominance/luminance YUV (YCrCb), chaque pixel est défini en fonction de sa l'intensité de sa couleur et de deux informations de sa chrominance (couleur).

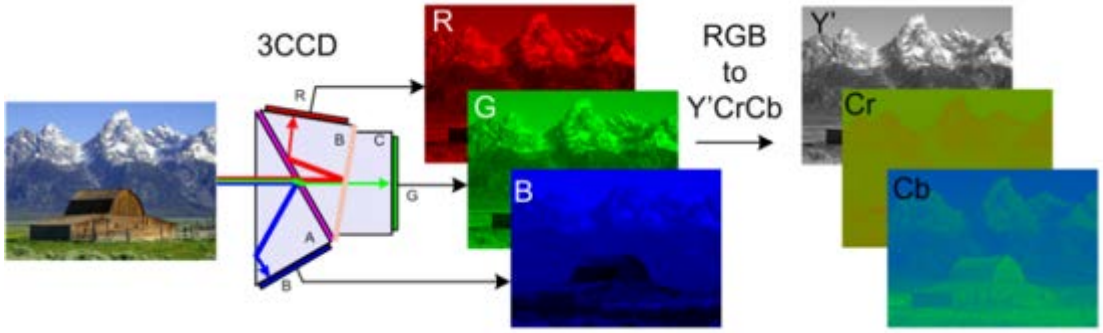


Figure 8 : Etape du passage RGB vers YUV

2) **Sous-échantillonnage :** Réduction de l'information occupée par la chrominance (couleur).

L'information perdue passera inaperçu car la vision humaine présente une sensibilité moindre à la couleur qu'à la luminosité.

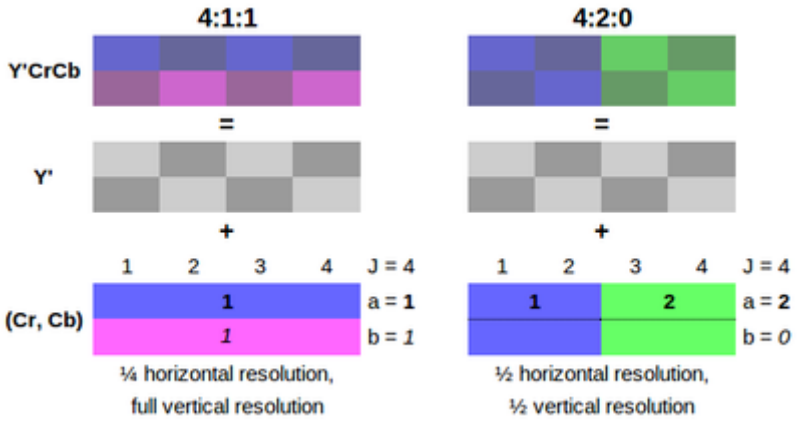


Figure 9 : Deux exemples de sous-échantillonnage

3) **Découpage en blocs :** La donnée va être divisé en macro-blocs, chaque macro-bloc est représenté sous forme de matrice de 8x8 pixels.
4) **DCT (Transformée en cosinus discrète) :** Détermination des degrés des changements d'un pixel à l'autre pour identifier les hautes et basses fréquences.

   Les fréquences basses, très présentes dans une image, sont des zones unies où les couleurs sont proches les unes des autres. Par contre, les hautes fréquences sont des zones de contraste, de changement rapide dans les couleurs.

   La formule mathématique du DCT est appliquée à la matrice afin de rendre plus facile l'étape suivante.
5) **Quantification :** C'est ici que l'on réduit vraiment la quantité d'information de l'image mais l'œil humain distingue mal les zones à hautes fréquences (de contrastes) alors la quantification vient diminuer l'importance des hautes fréquences qui ont été mis en évidence par la DCT.
6) **Encodage entropique :** Cette dernière étape comporte le codage RLE suivit du codage de Huffman.

   Le codage RLE consiste à replacer les caractères répétés par leur nombre suivit du caractère en question.

   Le codage de Huffman code les valeurs numériques les plus fréquentes avec un nombre de bits faible et les valeurs les plus rares avec un nombre de bits plus élevé. Cette compression se fait sans pertes (les pertes sont faites lors du seuillage et de la quantification), ainsi nous obtenons un fichier compressé.

En ce qui concerne le décodage, le fichier compressé passe par l'inverse de toutes les étapes citées de la dernière jusqu'à la première.

2.2.3 Codecs et conteneurs :

Avant toute chose, il faut éviter toute confusion entre conteneur et codec.

Un **conteneur vidéo** est un fichier qui permet de stocker des flux audio et vidéo, de les combiner et de les synchroniser. On reconnait un conteneur via son extension, par exemple ".avi" ou ".mkv".

Il ne s'agit en fait que d'une boite permettant de rassembler : un ou plusieurs flux vidéo, un ou plusieurs flux audio, des sous-titres, des éléments de chapitrage (comme les DVD), des métadonnées tel que le titre du média, le nom du réalisateur, la date, etc., une description des flux que contient le conteneur, ainsi que d'éventuelles autres données. Pour décompresser ces flux vidéos, les logiciels permettant de décompresser ou décoder ces flux sont les codecs.

Un **codec** est donc un logiciel conçu pour coder-décoder un flux de données numérique, il met notamment en œuvre les procédés formalisés par le MPEG. Il se caractérisent par : une architecture (une plateforme logicielle dite le 'Player'), par la définition des résolutions et d'un espace colorimétrique (une gamme définie de couleurs tel que le RGB).

Un fichier conteneur peut utiliser différents codecs, par exemple, un ".mkv" peut être encodé en format AVC ou DIVX. Et pour un même codec, il existe différentes versions.

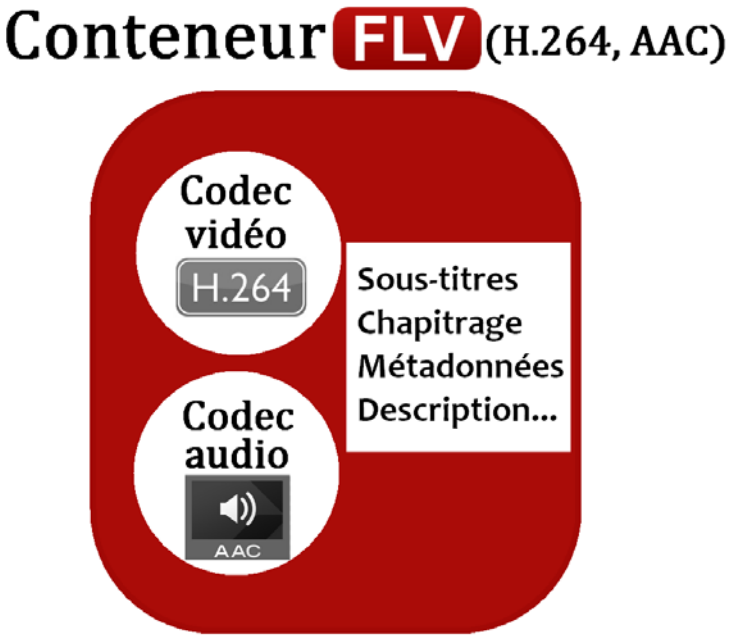


Figure 10 : Exemple d'un conteneur FLV supportant un codec vidéo H.264

Voici une liste de quelques conteneurs et leurs codecs supportés :

Conteneurs vidéo	Codecs vidéo
FLV FMP4 TS	H.264
MP4 3GPP	H.264, MPEG4
MPG MXF	MPEG2
WEBM	VP8, VP9
MPEG	H.264, MPEG1VIDEO, MPEG2VIDEO
MPEGTS	H.264, MPEG2VIDEO, VC1
MKV	H.264 , MPEG4, MJPEG, MPEG2VIDEO, VC1
AVI	H.264, MPEG4, MJPEG
ASF	WMV3, WMV3, VC1

Tableau 1 : Liste de conteneurs et leurs codecs supportés.

2.2.4 Qualité de service :

Il est assez important que le flux vidéo soit fluide lors de son visionnement. En terme de qualité de service (QdS) qui représente la performance du réseau.

Elle se caractérise principalement par un débit, un délai en milliseconde qui représente la latence, une gigue et aussi par le nombre de paquets perdus lors d'une transmission [15].

La Qds est déterminée par un seuil de tolérance par le nombre de paquets perdus par transmission et si le système renvoie beaucoup de paquets perdus, cela risque de créer une latence importante.

En plus la latence dépendra aussi du délai de propagation et celui de transmission d'un paquet d'une source vers une destination, qui pourrait être ralentie au niveau de chaque routeur à cause de leurs files d'attentes.

Un problème de dés-ordonnancement s'ajoute, il peut engendrer un gigue (la différence entre le temps d'arrivée de deux paquets qui se suivent dans un même flux) qui aura pour conséquence une désynchronisation des paquets à la réception. Pour assurer l'ordonnancement (Gérer l'ordre de départ des paquets) on utilise des algorithmes d'ordonnancement comme FIFO (Premier arrivé premier servi).

2.3 Réseaux :

2.3.1 Architectures réseaux pour une transmission vidéo :

1) Client-serveur :

L'architecture client-serveur se caractérise par la présence d'un serveur central multimédia qui relie tous les nœuds du réseau, il est donc le point central ce qui permet de mieux administrer le réseau, la problématique avec cette architecture c'est qu'en cas de panne du serveur central, la communication au sein du réseau sera interrompue.

Ce type d'architecture peut être intéressant quand il s'agit de partager de la vidéo à une vaste gamme d'utilisateurs.

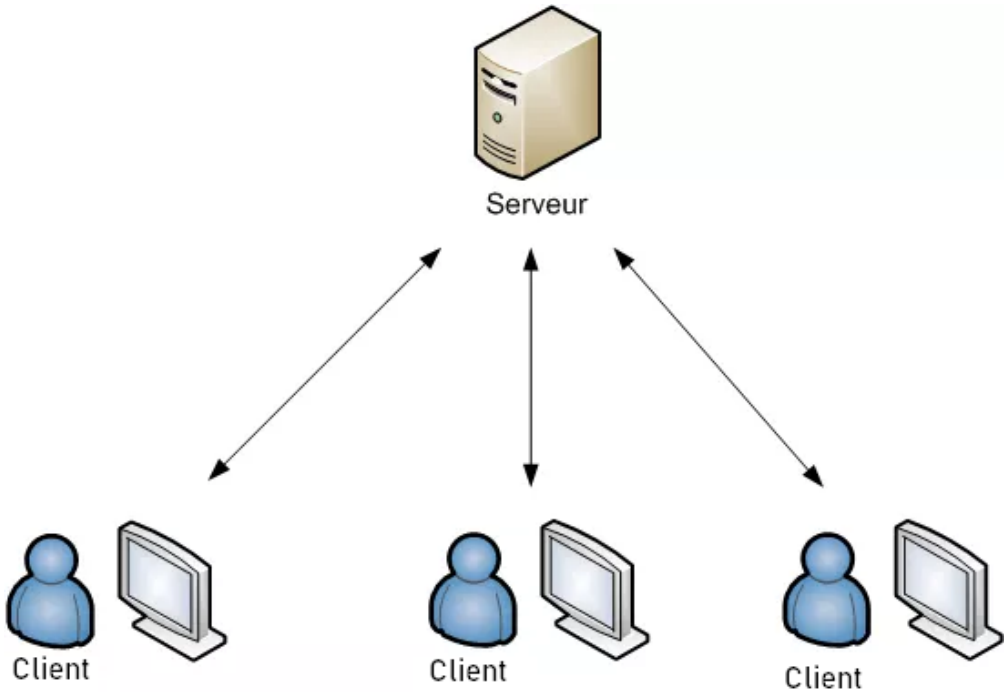


Figure 11 : Architecture Client/Serveur

2) Pair à pair :

L'architecture Pair à pair considère que toute les machines sur les réseaux sont égales, elles sont toutes client et au même moment serveur, l'avantage si une machine tombe en panne cela n'affectera pas le fonctionnement global mais le problème c'est que la sécurité dans ce genre d'architecture n'est pas garantie.

On peut utiliser du pair à pair dans le cas d'une communication vidéo temps réel entre deux personnes.

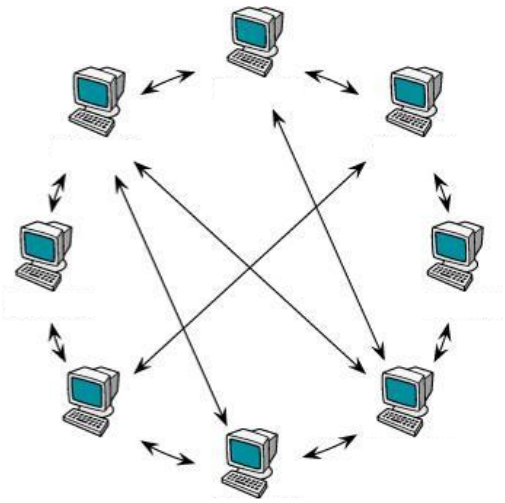


Figure 12 : Architecture Pair à Pair

3) Hybride :

Avec l'architecture hybride les clients n'ont pas de point central lors de l'initialisation, et font une élection pour élire un client parmi eux, qui deviendra un Client/Serveur ainsi l'avantage sera que dès que le serveur tombe en panne, les autres clients vont élire un autre mais au dépit d'une perte de connexion lors de l'élection et les machines participantes doivent être en mesure de jouer le rôle d'un serveur en ayant les capacités nécessaires pour supporter le flux qui circule dans le réseau.

2.3.2 Techniques de communications :

1) **Unicast :** qui désigne une transmission d'un nœud vers un autre nœud du réseau (une transmission point à point).
2) **Broadcast :** La transmission des paquets se fait d'un nœud vers le reste des nœuds du réseau.
3) **Multicast :** L'envoi de paquet d'un nœud vers un sous ensemble de nœuds du réseau.

2.3.3 Protocoles de communication :

2.3.3.1 Protocoles de transmission classiques :

1) TCP :

Transmission Control Protocol (TCP) est un protocole orienté connexion, c'est un protocole fiable, car pour communiquer il procède à l'ouverture de connexion, ensuite à chaque paquet envoyé, il reçoit un accusé de réception, et à la fin de la communication il ferme la connexion.

2) UDP :

Le protocole UDP (User Datagram Protocol) orienté « hors connexion », contrairement au TCP, il ne permet pas à l'émetteur de vérifier si les données sont effectivement reçues ou pas. Sa structure est plus simple et les transferts plus rapides. Par contre, il utilise aussi 65536 ports différents pour communiquer (de 0 à 65535), chaque application réseau utilise un ou plusieurs ports.

3) R-UDP :

Reliable User Datagram Protocol (R UDP) est un protocole de couche transport, il envoie les paquets avec un ordre garanti sans pour autant ajouter le complexité du protocole TCP, il n'ouvre pas de connexion comme le TCP mais par contre reçoit des accusés de réception et retransmet les paquets perdus (dans le cas où on ne reçoit pas d'accusé).

4) RPC :

Remote procedure call (RPC) est un protocole d'appel de fonction sur des machines distantes, grâce à ce protocole les utilisateurs peuvent interagir et s'envoyer des instructions et des données à travers le réseau.

5) RDP :

Remote Desktop Protocol (RDP) est un protocole de transmission développé par *Microsoft*, qui est régulièrement utilisé pour faire communiquer un client léger, tel qu'un terminal, avec un serveur centralisé.
Il repose sur le protocole T.120 utilisé pour la vidéo conférence, et permet de faire de l'administration à distance.

6) Web Socket :

Web Socket st un protocole réseau web qui permet la création de canaux de communication full-duplex par-dessus une connexion TCP mais il est en cours de standardisation par le W3C.

7) Web Socket Secure :

Web Socket Secure ajout la notion de sécurité au protocole web socket classique en utilisant des certificats SSL.

2.3.3.2 Protocoles de transmission temps réels :

RTP et RTCP sont deux protocoles situés au-dessus d'UDP, adaptés au besoin temps réel [16].

a. **Realtime Transport Protocol (RTP)** est conçu pour transporter les données, il reconstitue l'ordre des paquets, et détecte la perte de paquets
b. **RealtimeTransport Control Protocole (RTCP)** conçu pour échanger des messages de contrôle, supporte les communications multipoint. Il s'occupe aussi de la reconstitution du flux, l'identification du type de l'information transportée, et il contrôle l'arrivée des paquets à destination.
c. **RealTime Streaming Protocol (RTSP)**, c'est un protocole client-serveur situé au niveau applicatif il utilise le port 554, il permet de contrôler la distribution des flux RTP, offre des possibilités de contrôles pour avancer, rembobiner, pause dans le média, et il s'appuie sur RTP-RTCP pour la diffusion.

2.4 Conclusion :

Le réseau internet n'est malheureusement pas sans limitations pour transmettre des contenus multimédias, on doit faire face à certaines contraintes sur le réseau afin que la qualité d'usage des services multimédias soit acceptable, tel que :

Les pertes de paquets, la variation des délais qui sont des changements imprévisibles généralement dus à une congestion de paquets au niveau des routeurs comme nous l'avons expliqué auparavant ou à cause de la rupture d'un lien. La contrainte temps réel est souvent exigée par certaines applications se trouve perturbée par une QdS non garantie, il est difficile de satisfaire cette dernière car elle nécessite un délai assez court même avec l'amélioration de l'infrastructure du réseau depuis quelques années, le délai et la gigue restent les défis majeurs à surmonter.

Une dernière contrainte s’ajoute, c’est celle de la bande passante, car aujourd’hui les internautes sont de plus en plus exigeants, et pour pouvoir profiter de celle-ci dans une communication vidéo, il faut veiller à ce que l’image elle soit la plus compacte possible en utilisant des techniques de compression et de décompression, de plus, il est important que ces techniques puissent s'adapter à une vaste gamme de récepteurs, que l’appareil réceptif soit un ordinateur personnel, une tablette ou un smartphone car leurs performances et ressources sont très différentes.

Pour une communication vidéo, un protocole doit gérer le séquencement de paquets, l’identification des participants, et la surveillance de l’état de connexion. Un protocole tel que TCP pourrait faire l’affaire jusqu’ici, mais il est inadapté pour gérer la contrainte temps réel, il favorise la fiabilité au dépend du délai, et il existe seulement en version unicast.

UDP lui seul aussi ne peut satisfaire les besoins du temps réel, car il possède un service de transport non fiable, il n’a pas de connaissance de taux de perte et ne synchronise pas les paquets.

Beaucoup d'applications se basent sur le protocole RTP pour véhiculer des données vidéo sur Internet car il est adapté au données transmises en temps réel. Mais encore faut-il satisfaire le concept de E-maintenance et la réalité augmentée.

Chapitre 3

3 Conception d’une application de e-maintenance

3.1 Introduction :

Dans ce 3e chapitre, nous nous intéressons à la conception d’une application de e-maintenance industrielle associant un ensemble d’outils d’aide au diagnostic et à la réparation, elle est conçue pour être utilisée par au moins deux personnes, un technicien sur terrain et un expert qui généralement sera sur son bureau, ce dernier doit guider le technicien pendant la procédure de maintenance sur une machine industrielle.

Le technicien doit donc filmer la scène avec une caméra et la partager à l’expert à travers le réseau (qu’il soit dans le même secteur ou à des centaines de kilomètres) pour qu’il puisse l’assister en temps réel.

L’expert doit voir la scène exactement telle qu’elle est réellement, il doit visualiser la scène reçue par le technicien et doit avoir sur son écran une multitude d’outils virtuels 3D (augmentés) qu’il peut déplacer à sa guise et les afficher en temps réel sur l’écran du technicien pour le guider lors de la procédure de maintenance, il peut aussi disposer d’un chat écrit et un chat vocal afin de renforcer le système e-maintenance, nous devons donc veiller à avoir une communication vidéo dans un sens, et gérer le contrôle des objets de réalité augmentée dans l’autres sens, et synchroniser la communication entre les utilisateurs.

Nous devons donc réaliser un système de réalité augmentée collaboratif pour la e-maintenance, il doit donc permettre à un ou plusieurs techniciens dans un domaine quelconque de se faire assisté par un expert dans le domaine à distance (par internet) ou dans le même secteur (en local).

3.2 Système de e-maintenance réalisé avec le protocole RDP :

Cette version permet à un utilisateur de prendre le contrôle total de l'application d’un utilisateur distant, nous utilisons pour cela le protocole RDP (Remote Desktop Protocol), l’architecture du système est de type client-serveur.

Nous avons donc un serveur RDP, il permet d’avoir le contrôle l’application du client RDP, dans notre cas, le serveur sera l’expert, et le client le technicien, de cette manière l’expert pourra contrôler des objets 3D (réalité augmentée) présents dans l’application du technicien, cette dernière affichera la scène filmée et les objets 3D.

RDP se charge de l'envoie des frames, compression utilisant un codec H.264/AVC 444 [17] dans la version RDP 10, ainsi que du chiffrement des données car la connexion se fait avec une clé hachée.

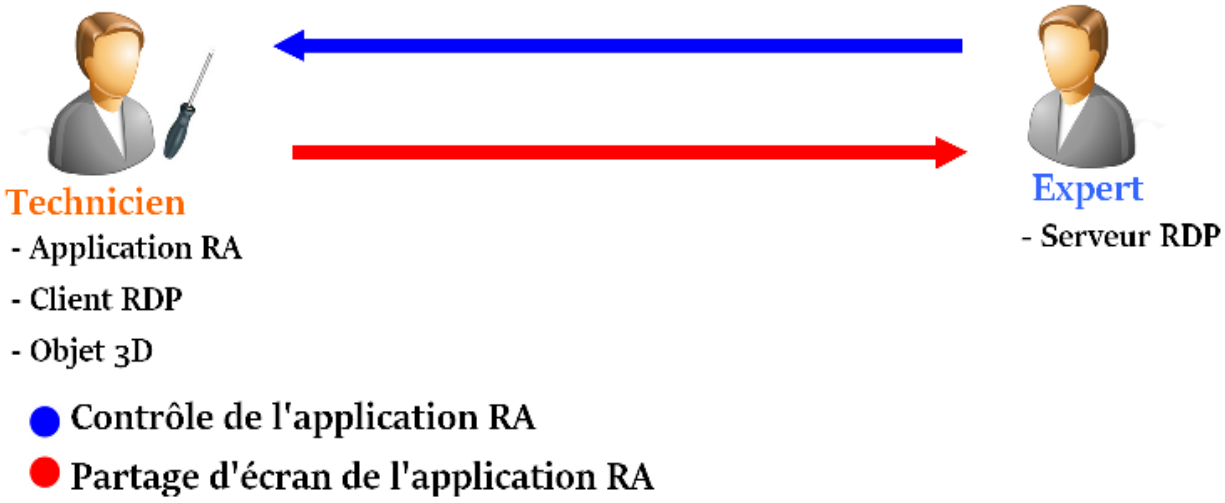


Figure 13 : Schéma - Architecture client-serveur RDP

Nous avons ici deux applications :

3.2.1 L'application de l'expert :

C'est un serveur RDP, il se charge d'afficher l'écran du novice et de donner un contrôle total de son application, il est réalisé avec Visual Studio et donc réalisé en dehors du cadre d'Unity.

Par mesure de sécurité, le serveur RDP génère une clé de connexion qui devra être communiquée au client RDP afin qu'il puisse se connecter au serveur RDP, cette clé contient l'adresse IP privée et publique du serveur, les ports sur lesquels il écoute et d'autre informations, toutes les informations sont hachées avec le hash SHA256.

3.2.2 L'application tu technicien :

Elle contient deux applications :

- Une application réalisée avec Unity pour gérer le côté RA (objets 3D), ayant un écran affichant la scène filmée par la caméra en temps réel.
- Un client RDP réalisé avec Visual Studio, il s'occupe du partage d'écran de l'application, il a la possibilité d'encoder les images à transmettre avec un codec H.264/AVC 444, le client s'exécute en arrière-plan après que l'utilisateur ait introduit la clé de connexion générée par le serveur RDP.

Une fois le client RDP connecté, l'écran est partagé en temps réel en donnant l'accès total sur l'application du technicien, ce qui permet à l'expert d'avoir un contrôle sur les objets 3D.

3.2.2 Avantages et inconvénients :

Le concept du partage vidéo et le contrôle en temps réel est parfaitement respecté, mais le serveur RDP donne beaucoup trop de contrôle à l'expert sur l'application du technicien, il peut fermer son application RA par erreur par exemple ou si le technicien reçoit une notification au moment où il fait la maintenance celle-ci pourrait être vue par l'expert et cela pose un grand problème de confidentialité, un autre inconvénient s'ajoute à ce dernier, c'est la communication de la clé qui doit être communiquée avant la connexion, et être introduite manuellement.

3.3 Système utilisant les protocoles http, TCP et un codec MJPEG :

Ce système a une architecture de type client-serveur, mais il est assez complexe, nous avons utilisé un codec MJPEG grâce une bibliothèque externe (en dehors du cadre d'Unity) prête à être utilisée et programmée en C#.

Nous disposons de deux entités principales :

3.3.1 L'application de l'expert :

Elle se compose de 2 programmes :

- **Un serveur UDP** ayant une adresse IP et un port :7777, ce serveur-là prend en charge le contrôle et l'interaction des objets 3D, il est réalisé avec le logiciel Unity 3D.
- **Un client TCP-http** du serveur appartenant au technicien, cette partie cliente permet d'afficher l'écran du technicien novice chez l'expert, grâce à l'adresse IP et le numéro de port du serveur TCP-http, sur lequel il se connecte en TCP en entrant (adresse_ip : port), et récupère les données sur Unity en http grâce la classe WWW en c#.

3.3.2 L'application du novice :

Elle se compose aussi de deux programmes :

- **Un client UDP**, qui se connecte au serveur UDP de l'application de l'expert grâce à l'adresse IP et au port de ce dernier, cette connectivité permet aux objets du novice d'être contrôlés par l'expert (réalité augmentée), réalisé avec Unity.
- **Un serveur web** avec une ouverture de connexion TCP sur le port 8080, le partage de son écran après l'avoir encodé en MJPEG se fait via le protocole http, le lancement de ce serveur est totalement transparent, il se lance en arrière-plan automatiquement avec

le lancement de l'application de réalité augmentée (client UDP) il s'exécute en arrière-plan sans apparaitre dans la barre des tâches, nous l'avons réalisé avec Visuel Studio car la bibliothèque est programmée hors du moteur Unity.

3.3.3 Avantages et inconvénients :

Le fait d'utiliser le codec MJPEG facilite longuement la tâche en codant les images du flux vidéos chacune en format JPEG, mais en plus du fait que le système n'est pas sécurisé, n'importe qui peut visualiser le flux en tapant l'adresse IP publique du serveur http sur un navigateur, il est aussi lourd car nous faisons appel à chaque fois de l'application réalisée avec Unity vu que nous avons intégré une bibliothèque externe à un système réalisé avec Unity.

3.4 Systèmes Système réalisé avec Photon :

3.4.1 Architecture de communication :

L'architecture de notre système est de type client-serveur, nous utilisons Photon qui est un serveur dédié offrant une large bande passante et pouvant supporter jusqu'à 20 utilisateurs à la fois, nous aurons besoins que de deux utilisateurs pour chaque opération de maintenance.

Toutes les données envoyées par les utilisateurs transitent par le serveur, et ceci grâce à des objets qui sont instanciés dans le réseau, chaque objet a son propre *PhotonView* servant à identifier de façon unique son utilisateur.

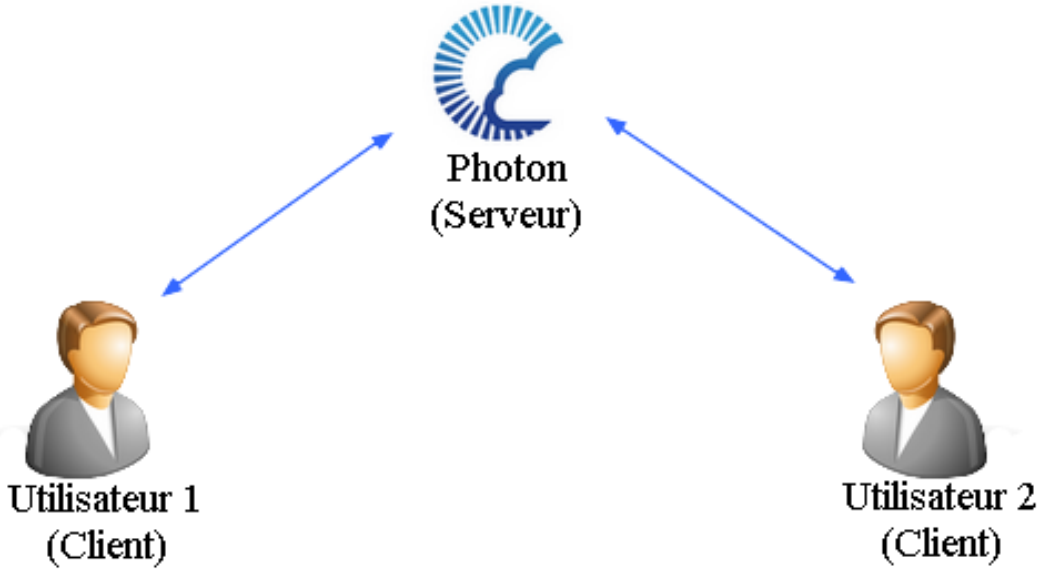


Figure 14 : Schéma - Architecture client/serveur (photon)

3.4.2 Protocoles de communication :

a. Protocole R-UDP :

Nous avons utilisé en premier lieu le protocole R-UDP afin de réduire les délais de transmission, car ce dernier n'attend d'accusés de réception lors de l'envoi de paquets.

Après plusieurs tests avec R-UDP, nous avons remarqué que le serveur finit par déconnecter les clients après quelques minutes de connexion, nous avons conclu que le serveur subit lors de la communication un déni de service car il retransmet les paquets perdus à chaque fois, et donc les clients sont déconnectés au bout de quelques minutes.

b. Protocole WebSocket :

Nous avons aussi testé le protocole WebSocket, mais aussi vite nous l'avons exclu de la liste des choix des protocoles pour le transfert vidéo car son utilisation provoque une latence assez conséquente et ne permet pas de garantir une bonne synchronisation.

c. Protocole TCP :

Etant donné que le serveur Photon ne supportait pas la transmission vidéo par le protocole R-UDP, nous avons opté pour le protocole TCP, n'étant pas le protocole idéal pour une communication vidéo en temps réel, car il attend à chaque envoie un accusé de réception, celui-ci convient parfaitement au serveur qui n'est pas surchargé de paquets et ne déconnecte donc pas les clients.

Pour réduire la latence de l'envoi vidéo, nous nous sommes concentré sur la méthode de compression avant l'envoie et limiter le nombre de frames envoyés par seconde, car pour une application de e-maintenance, on n'a pas forcément besoin de 24 frames par secondes au minimum comme pour une visio-conférence par exemple, on peut se permettre d'envoyer une 20aine de frames/seconde.

d. Protocole RPC :

Nous utilisons le protocole RPC pour appeler des fonctions sur les machines distantes, il nous permet d'envoyer des commandes RPC pouvant contenir différentes données, selon le besoin de l'utilisateur.

Pour définir une méthode RPC, il faut rajouter l'attribut [PunRPC] puis définir la méthode comme nous définissons n'importe quelle autre méthode.

Pour appeler la fonction distante, on a besoin de spécifier le **PhotonView** pour identifier l'utilisateur.

Un appel se fait de cette façon :

PhotonView.RPC (''nom de la méthode'', destinataire, ''option variable'', ''autre option si besoin'') ;

3.4.3 Principe de fonctionnement dans Photon :

Pour un système de e-maintenance, deux ou plusieurs entités principales doivent être mises en place, dans notre cas, il y aura deux types d'utilisateurs principales, ce sont les clients qui se connecteront au serveur Photon.

Les clients ne communiquent pas entre eux s'ils ne sont pas connectés dans une room (voir figure 18), donc pour qu'un utilisateur se connecte au serveur, il doit d'abord rejoindre une room, une room sert principalement à isoler un groupe de personne d'autre groupe d'utilisateurs dans le cas où plusieurs doivent se connecter au même serveur mais séparément, ainsi les messages sur le serveur ne circuleront pas à tort et à travers sur le réseau mais seront bien organisés.

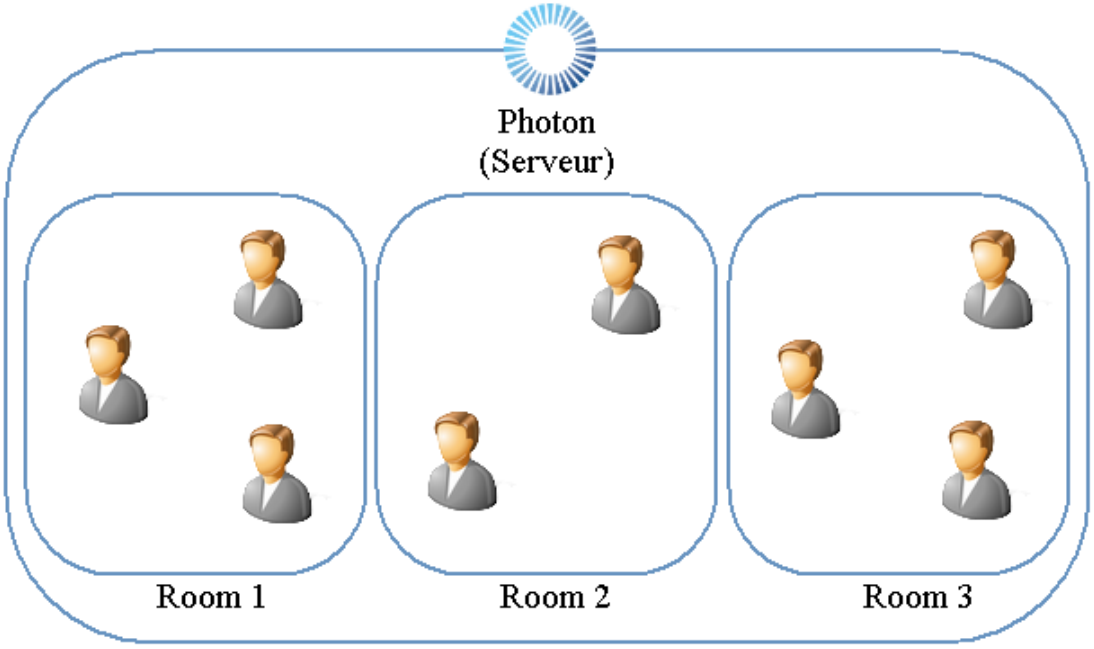


Figure 15 : Schéma démontrant la notion de Room

La communication est donc privée dans chaque Room, les utilisateurs connectés dans d'autres room n'ont pas accès aux informations qui circulent dans la room, ce qui rend la communication entre les utilisateurs d'une même room sécurisée.

Les deux utilisateurs principaux qui doivent se connecter à la room sont le technicien novice, qui doit filmer la scène, et l'expert pour l'assister durant l'opération de maintenance avec

différents outils, toutes les données passent par le serveur comme le démontre la figure ci-dessous :

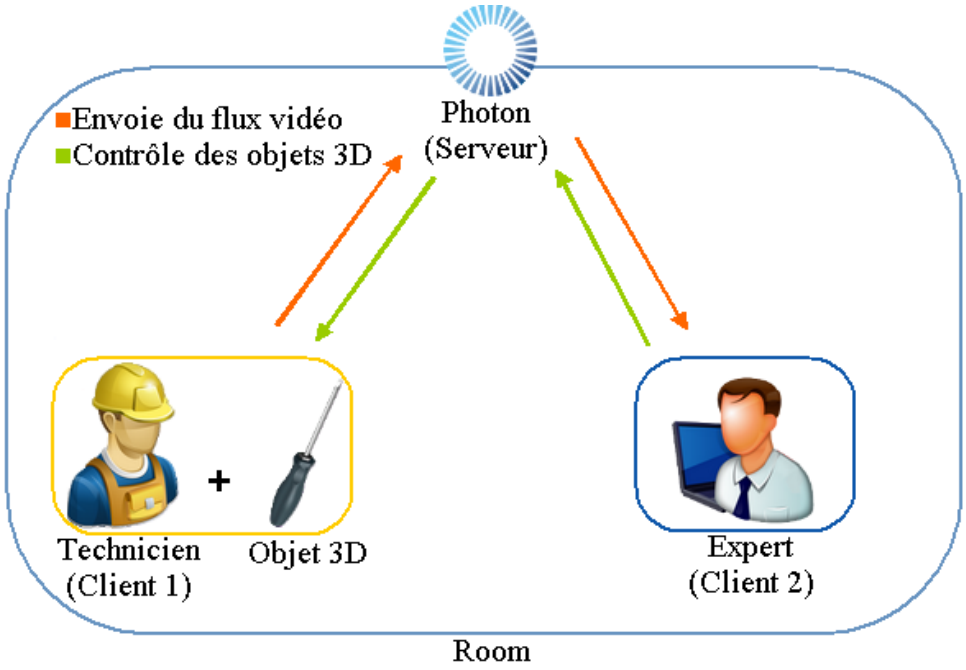


Figure 16 : Schéma du fonctionnement du système réalisé avec Photon

Nous avons donc conçu deux applications différentes :

a. **L'application du novice (client) :** L'application a pour rôle d'envoyer les images de la scène au host en lui envoyant des appels *RPC* au serveur bien sûr après avoir compresser les frames, elle doit aussi se charger d'exécuter les traitements des objets 3D de la réalité augmentée. Contrairement à la version d'UDP, celle-ci donne plus de transparence à l'utilisateur, car il n'a ni d'adresse IP à saisir, ni port pour se connecter au serveur, tout est géré par le serveur Photon grâce au *NetworkIdentity* de chaque utilisateur.

b. **L'application de l'expert (client) :** L'application doit permettre à l'expert de guider le technicien grâce aux objets 3D, indications etc… et ceci en envoyant des appels RPC au serveur, et comme n'importe autre client de Photon, dès son lancement, il se connecte au serveur.

3.4.4 Sécurité :

L'utilisation de room dans Photon procure déjà un certain niveau de sécurité, l'ajout d'une base de donnée qui permet d'identifier puis authentifier les utilisateurs viens complémenter cela, et les messages entre le client et le serveur sont chiffré par défaut par Photon, et offre davantage de sécurité avec différentes fonctionnalités comme la vérification des certificats

SSL si l'on utiliser WebSocket par exemple, nous allons donc aborder quelques notions de sécurité que nous utilisons à travers Photon pour renforcer la sécurité du système.

3.4.4.1 Diffie Hellmann :

Diffie Hellmann a été créé par Whitfield Diffie et Martin Hellmann, il était souvent utilisé par les internautes pour remplacer le chiffrement RSA qui était breveté et donc non utilisable librement aux états unis jusqu'à l'an 2002.

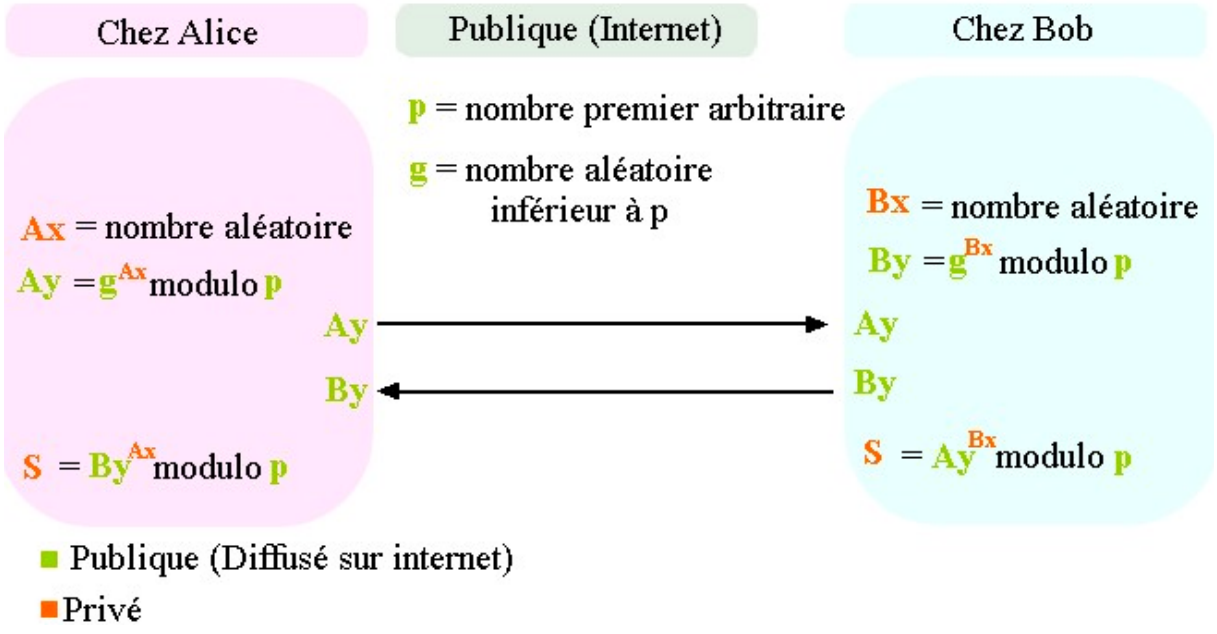


Figure 17 : Schéma décrivant la méthode de Diffie Hellman [23]

C'est une méthode d'échange de clé qui permet à deux entité de se mettre d'accord sur un chiffre qui leurs servira à chiffrer leurs conversations afin que leur communication ne soit pas interceptée par une personne malveillante qui se met au milieux de leur communication (Man in the middle).

3.4.4.2 Cryptage AES :

AES (Advanced Encryption Standard) est un algorithme de chiffrement symétrique, il offre des tailles de blocs et de clés compris entre 128 et 256 bits, il remplace l'algorithme DES qui n'utilise que 56 bits.

Les différentes opérations sont répétées plusieurs fois et à chaque répétition, une clé unique est calculée à partir de la clé de cryptage et elle est intégrée aux calculs.

L'AES peut être cracker mais cracker une clé de 128 bits avec un ordinateur prendrais des millions d'années.

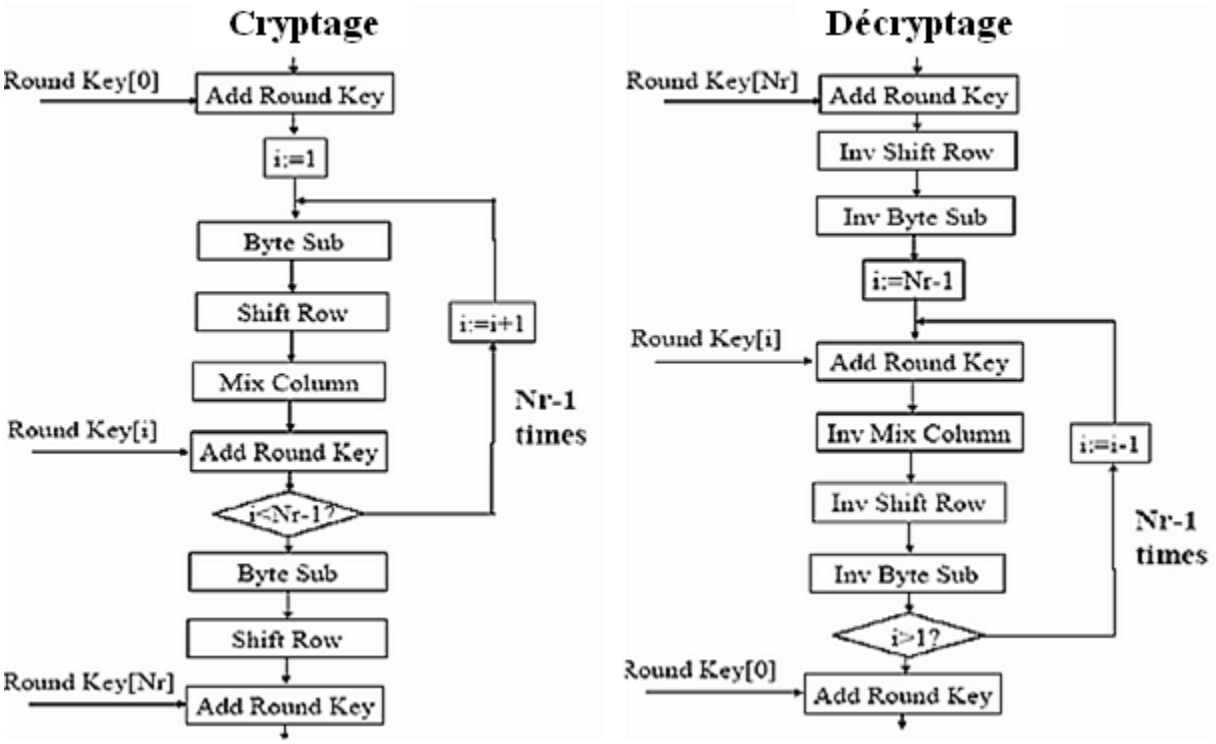


Figure 18 : Schéma des étapes du cryptage AES [21]

3.4.4.3 PhotonSecure :

L'API de Photon échange automatiquement les clés (une clé de 160 bits) de chiffrement entre le client et le serveur dès que le client se connecte, à l'aide de l'algorithme Diffie Hellmann. Cette clé sera utilisée pour le cryptage AES [22].

Le serveur Photon émet un Jeton (*Token*), un résumé chiffré des valeurs d'authentification du client, le client n'a pas besoin de lire le *token* à son authentification.

Le serveur est la seule entité capable d'actualiser les jetons, et il le fait quand un utilisateur crée ou rejoins une room, le jeton a une durée de vie de 60 minute mais il est néanmoins actualisé à chaque fois qu'un utilisateur déclenche un évènement (envoie d'un RPC par exemple), donc si le jeton reste pendant 60 minute sans aucun évènement de la part de l'utilisateur le jeton expirera, et l'utilisateur devra se reconnecter à nouveau pour jouir de l'utilisation de ce jeton, sinon il aura un message d'erreur qui n'entraine ni l'arrêt de l'application ni sa déconnexion.

Dans notre application on a utilisé la fonction :

PhotonView.RpcSecure (''nom de la méthode'', destinataire, bool crypter, ''paramètre'') ;

Cette fonction chiffre les arguments (les données) de la méthode, qui sont envoyés aux autres clients et les données seront déchiffrées à leurs arrivée chez le destinataire.

3.4.5 Avantages et inconvénients :

Le serveur Photon Networking d'Unity permet d'utiliser plusieurs protocoles de communications, tel que TCP, Web Socket, http et R-UDP. En l'utilisant, nous bénéficions de certains privilèges liés à la sécurité des données, mais c'est un serveur externe qui exige la disposition du réseau internet, ce qui rend les clients dépendants de ce dernier, et ne donne pas le privilège de l'implémenter au CDTA, malgré le fait qu'il existe une version Self-Hosted de Photon qui permet de déployer son propre serveur Photon, afin de contrôler les clients et pleins d'autres fonctionnalités intéressantes, mais cette version est payante donc nous nous sommes contenté d'utiliser la version gratuite pour le moment.

3.5 Systèmes réalisés avec Unet :

Nous avons utilisé **Unet** (moteur de réseau d'Unity) pour réaliser trois architectures différentes, il permet de créer et de configurer un réseau en se basant sur des architectures client-serveur ou client-host.

Unet propose les protocoles réseau **UDP** et **RPC**, que nous utilisons pour les 3 architectures.

L'envoie des données à travers Unet se fait comme suit :

2. **Par des appels *RPC* :** Si l'envoie se fait du serveur à tous les clients y compris les hosts.
3. **Des appels *TargetRPC* :** Si l'envoie se fait du serveur à un client spécifique.
4. **Des *Command* :** Si l'envoie se fait du client au serveur (ou host)

3.5.1 Architecture client-serveur :

L'architecture choisie est de type client-serveur, où nous avons trois entités au lieu de deux, les applications des utilisateurs (Expert et novice) sont clientes, celles-ci ne peuvent pas communiquer entre elles sans le serveur qui sert de point central, les 2 clients se connectent au serveur grâce à son adresse IP privée (si le réseau est local) ou publique si le serveur est distant en utilisant le NAT (Network Address Translation) que nous détaillerons par la suite, et

communiquent à travers lui, la figure ci-dessous décrit l'architecture ainsi que la connexion des clients au serveur.

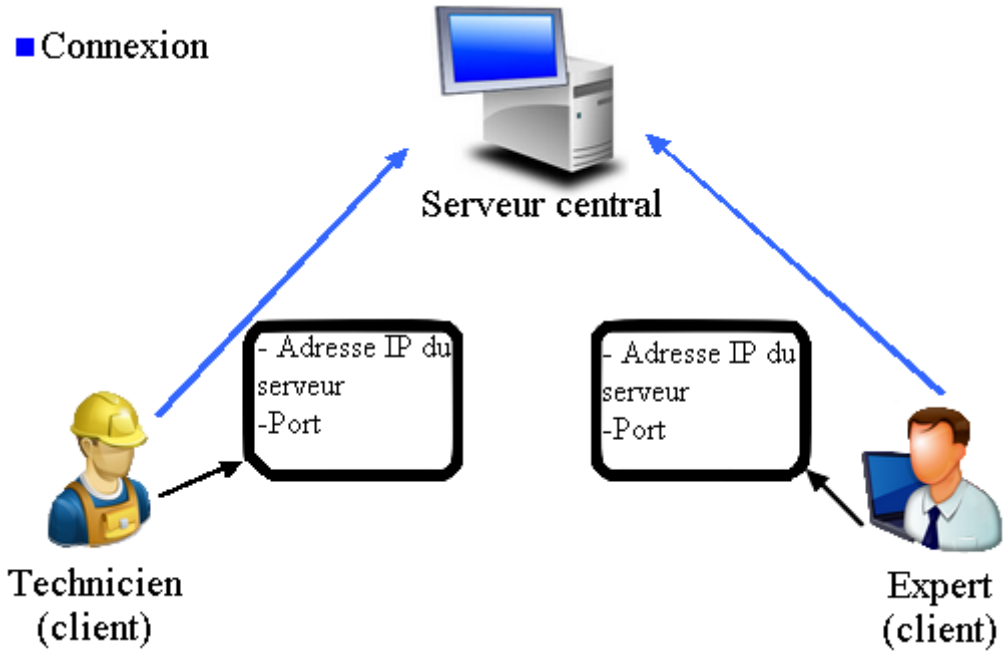


Figure 19 : Schéma démontrant la connexion dans l'architecture client-serveur

Pour pouvoir communiquer chaque client doit avoir son propre *NetworkIdentity* qui lui sera comme identifiant unique sur le réseau, celui-ci va être instancié dans le réseau dès que l'utilisateur se connecte au serveur en introduisant son adresse IP et son port. Le serveur disposera des *NetworkIdentity* de tous les clients connectés à lui ainsi il pourra différencier entre eux et saura donc distinguer les émetteurs des messages reçus.

Afin d'initier la connexion entre les clients, on doit passer une étape assez importante pour que le serveur puisse transmettre la donnée reçue, il doit associer à chaque *NetworkIdentity* d'un client le *NetworkIdentity* de l'autre client afin que chacun des deux clients puisse adresser son message à l'autre et non pas au serveur uniquement. Nous décrivons cette étape-là sur le schéma en dessous :

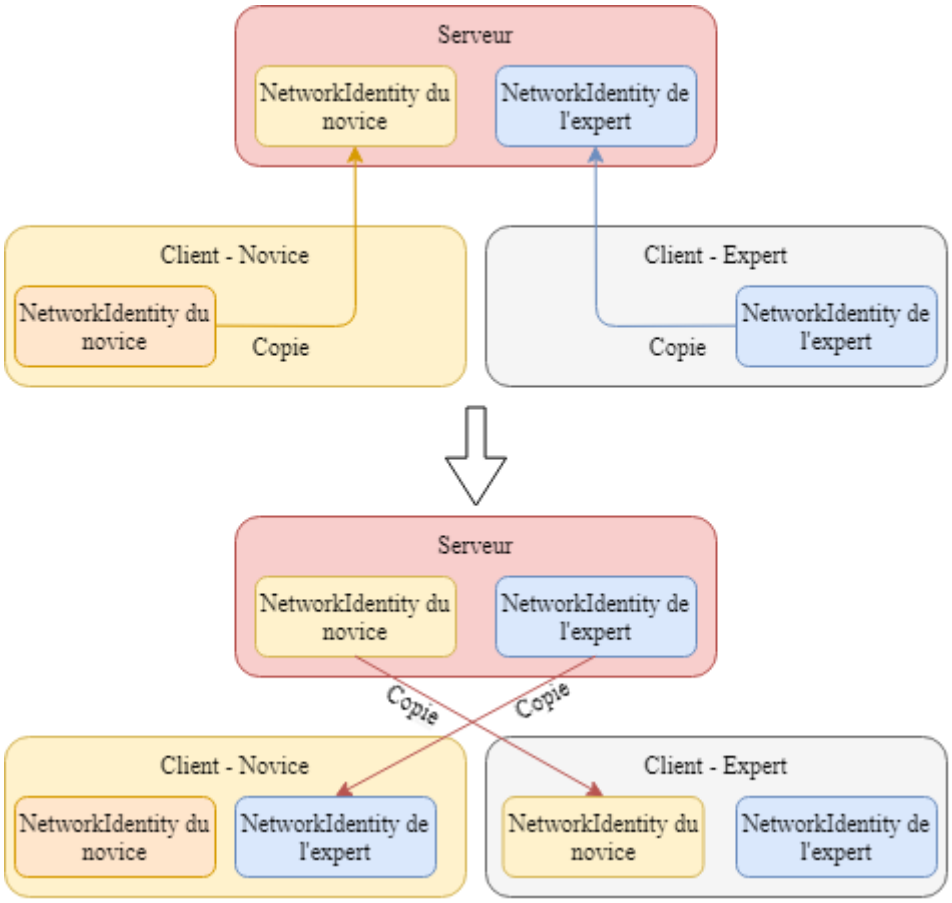


Figure 20 : Etapes de distribution des NetworkIdentity établies par le serveur

Une fois la connexion faite, les données peuvent enfin être envoyées et chaque application peut exécuter sa tâche.

Nous avons donc créé 3 applications :

3.5.1.1 **Application serveur :** Son rôle est le suivant :

- Lors de son lancement, l'utilisateur doit spécifier le port par lequel il recevra les connexions qui devra être communiquer aux clients avec son adresse IP.
- Identifier les *NetworkIdentity* des clients qui se connectent à lui et associe à chaque NetworkIdentity d'un client le NetworkIdentity de l'autre client comme décrit dans la **figure 13**.
- Recevoir des *Command* de la part de l'un de ses clients, et transfert la donnée reçue en envoyant des *TargetRPC* au client récepteur.

3.5.1.2 **Application du novice** : C'est un client du serveur, son rôle est comme suit :

- Une fois connecté au serveur, il active la caméra de l'appareil et commence à filmer, rajouter les augmentations, compresser les images et les envoyer au serveur grâce à des *Command* au serveur, ce dernier se chargera de les transférer à l'expert.

3.5.1.3 Application de l'expert : C'est un deuxième client du serveur, son rôle est le suivant :

- Une fois connecté au serveur, il reçoit directement les frames, il peut à présent contrôler l'objet distant en envoyant des *Command* au serveur qui les transmettra au technicien.

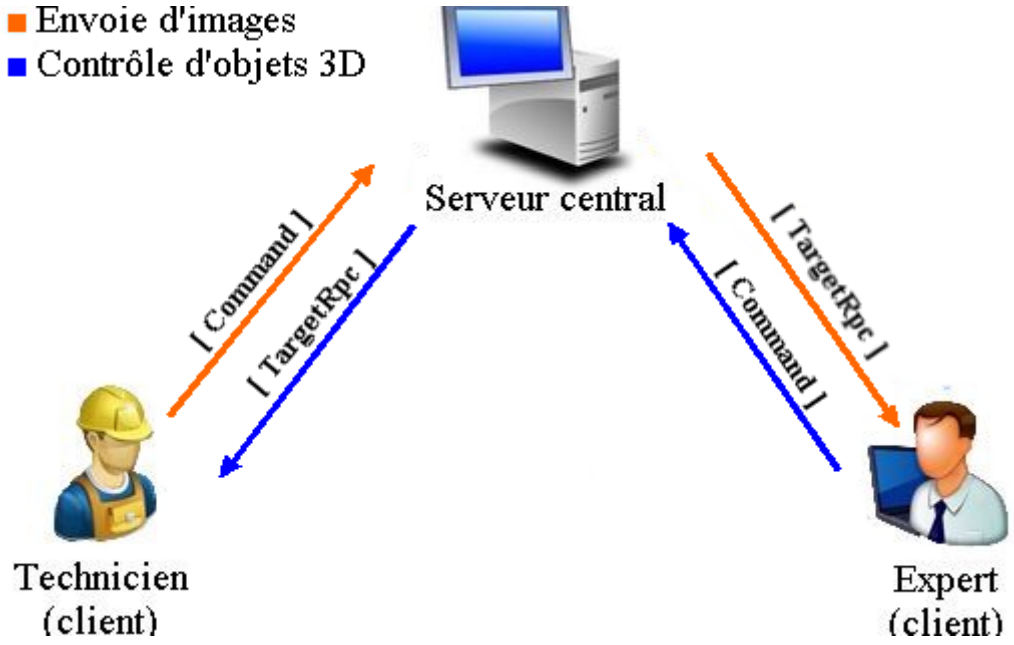


Figure 21 : Schéma d'échange de données dans une architecture client-serveur

3.5.1.4 Avantages et inconvénients :

Dans cette architecture toutes les données transitent par le serveur, ce n'est pas l'idéal pour une transmission vidéo, mais il n'est pas possible de faire autrement sachant que le serveur dans ce cas-là a pour but d'identifier les clients dans l'entreprise afin de noter l'heure ou la durée de la maintenance par exemple.

L'idéal aurait était de garder l'architecture client-serveur mais que le serveur puisse initier la connexion, et que le client puisse envoyer le flux vidéo directement au client sans passer par le serveur, mais ce n'est malheureusement pas faisable sur Unity.

3.5.2 Architecture Client-Host, Unet :

Nous avons réalisé l'architecture client-host afin d'éviter que les données transitent par un serveur intermédiaire, nous voulons les images de la scène passe du technicien à l'expert directement afin de réduire au maximum le délai de réception, donc nous aurons 2 utilisateurs principales, un client UDP, ce sera l'application du technicien, l'application de l'expert sera un host (client et serveur en même temps).

Le technicien doit disposer de l'adresse IP et du port de l'expert afin qu'il puisse s'y connecter comme décrit dans le schéma suivant :

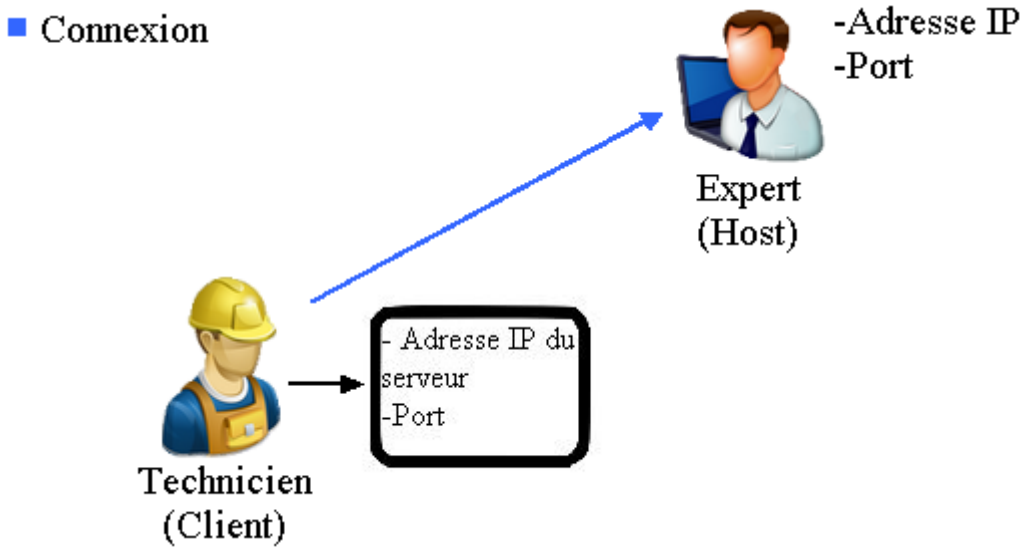


Figure 22 : Schéma décrivant l'architecture client-host

Nous avons créé 2 applications différentes :

3.5.2.1 L'application du novice (client) : L'application a pour rôle d'envoyer les images de la scène au host en lui envoyant des *Command* bien sûr après avoir compresser les frames, ainsi que d'exécuter les demandes du serveur, elle doit aussi se charger d'exécuter les traitements des objets 3D de la réalité augmentée.

3.5.2.2 L'application de l'expert (host) : C'est un client et serveur en même temps (host), il doit être lancé pour que le technicien puisse s'y connecter étant donné qu'il est considéré comme étant serveur du système, son rôle est de permettre à l'expert de guider le technicien grâce aux objets 3D, indications etc… et ceci en envoyant des *TargetRPC.*

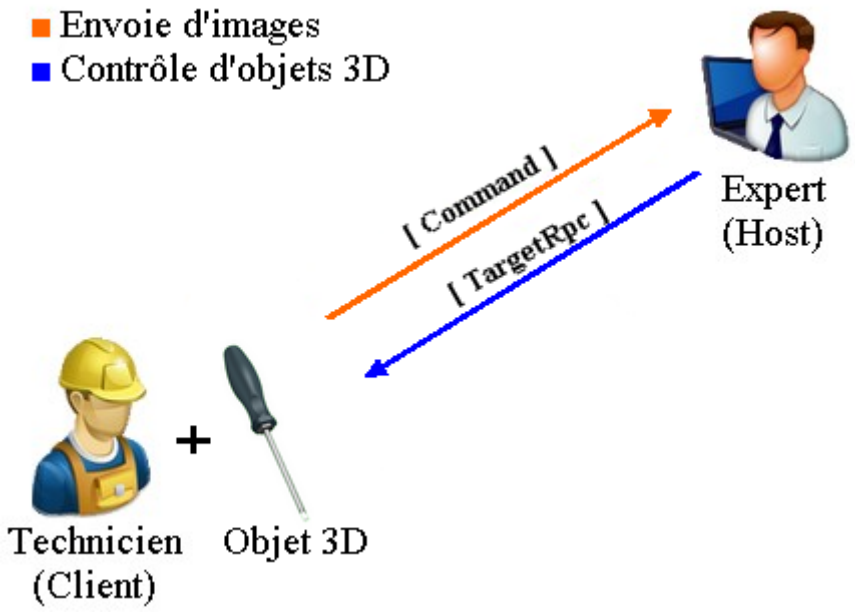


Figure 23 : Schéma d'échange de données dans une architecture Client Host

3.5.2.1 Avantages et inconvénients :

Dans cette architecture, les données passent directement d'un utilisateur à un autre, il n'y a pas de serveur intermédiaire, et donc la latence est minimisée, mais le manque d'un serveur qui doit être implémenté dans l'entreprise pour identifier les utilisateurs et l'heure de la maintenance n'arrange pas vraiment les administrateurs.

3.5.3 Système hybride (client-host) :

Après avoir réalisé 2 architectures différentes avec Unet, et que ces dernières ne répondent pas à nos attentes, nous voulons faire une combinaison de deux architectures, nous précisons que la nomination 'hybride' ici ne veut pas dire que l'architecture réseau est hybride, c'est juste une appellation lié à son fonctionnement, car nous voulons avoir un serveur qui sera implémenté au sein du CDTA, et en même temps diminuer la latence au maximum, donc nous avons conçu une architecture composée de 3 entités principales :

3.5.3.1 Host (Expert local) : Situé à l'entreprise, il jouera deux rôles différents :

- Une application pour un expert qui sera présent en local (dans l'entreprise), donc qui permettra de manipuler les objets 3D et de pouvoir les contrôler à distance pour assister le novice.
- Un serveur intermédiaire entre le client technicien et un client expert distant (hors de l'entreprise), pour transférer les données d'un client à un autre et aussi de contrôler les connexions dans le cas où plusieurs experts doivent assister à une opération de

maintenance ou dans le cas où un expert doit assister 2 techniciens en même temps, etc…

3.5.3.2 Client (Technicien novice) : pour le technicien, elle servira au novice pour filmer la scène et l'envoyer sous formes de frames après compression au host.

3.5.3.3 Client (Expert distant) : La 3ème application est presque la même que la première, donc conçue pour un expert qui sera distant (c'est-à-dire dans le cas où l'expert est hors de l'entreprise), mais celle-ci ne disposera pas des mêmes privilèges que la première application, car elle sera cliente de la première qui est un host expert qui jouera le rôle d'un serveur intermédiaire, elle doit donc se connecter à elle pour communiquer avec un client technicien.

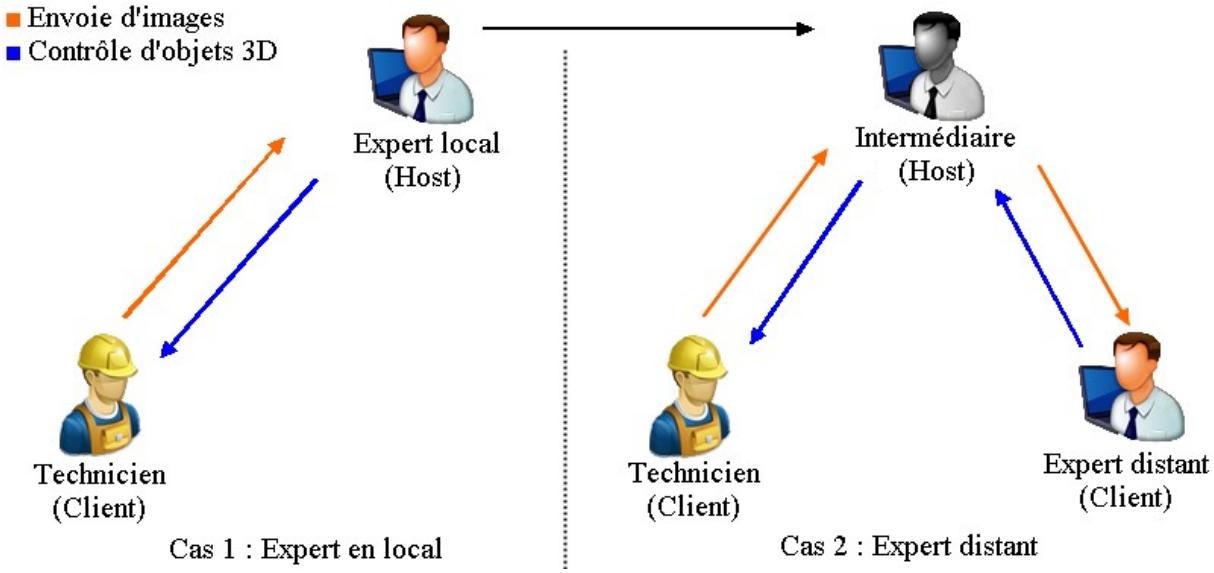


Figure 24 : Schéma décrivant les deux cas du système hybride

3.5.3.4 Distribution des rôles :

Nous avons deux acteurs principaux le technicien novice et l'expert, ainsi que deux cas, quand l'expert est local et quand il est distant, ci-dessous un diagramme décrivant les étapes de la procédure, il faut savoir que la 3ème et 4ème étape sont répétées jusqu'à la fin de l'opération de e-maintenance.

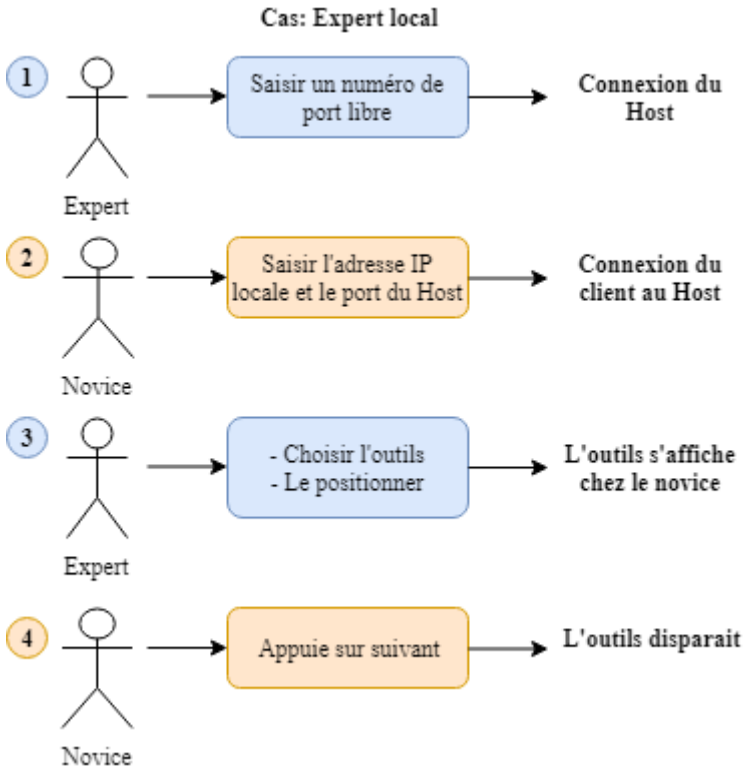


Figure 25 : Diagramme de distribution des rôles aux acteurs (Cas expert local)

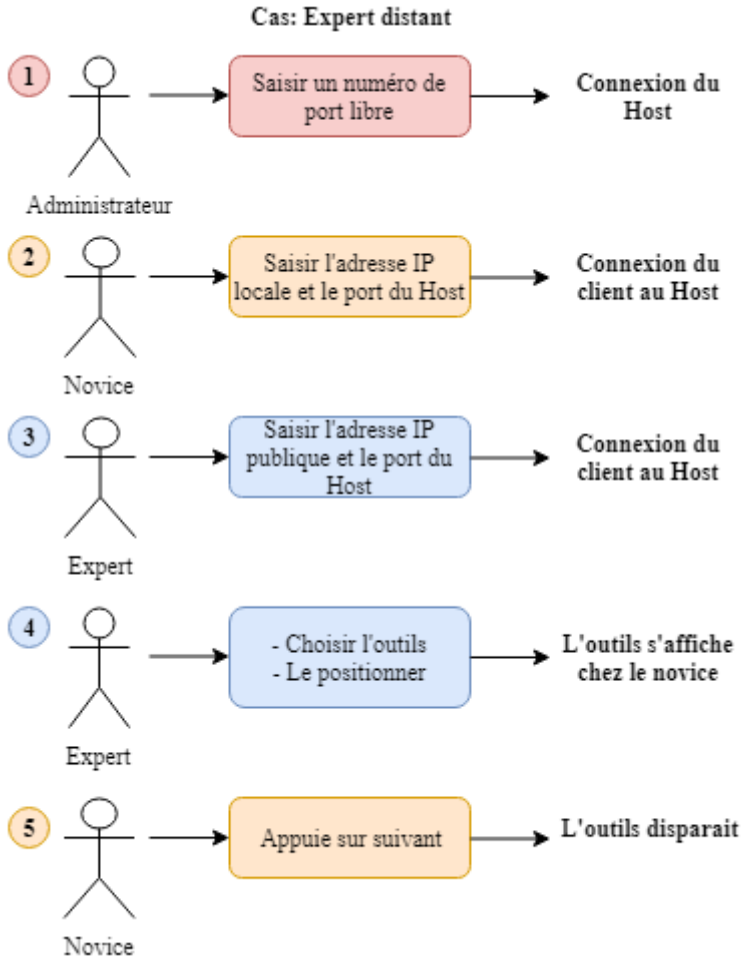


Figure 26 : Diagramme de distribution des rôles aux acteurs (Cas expert distant)

3.5.3.5 Ouverture de connexion :

Voici un diagramme qui détaille le déroulement de l'ouverture de connexion du host et des clients, le host doit se connecter en premier, les clients locaux doivent introduire **l'adresse IP locale** du host, le client distant, introduit **l'adresse IP publique** du host.

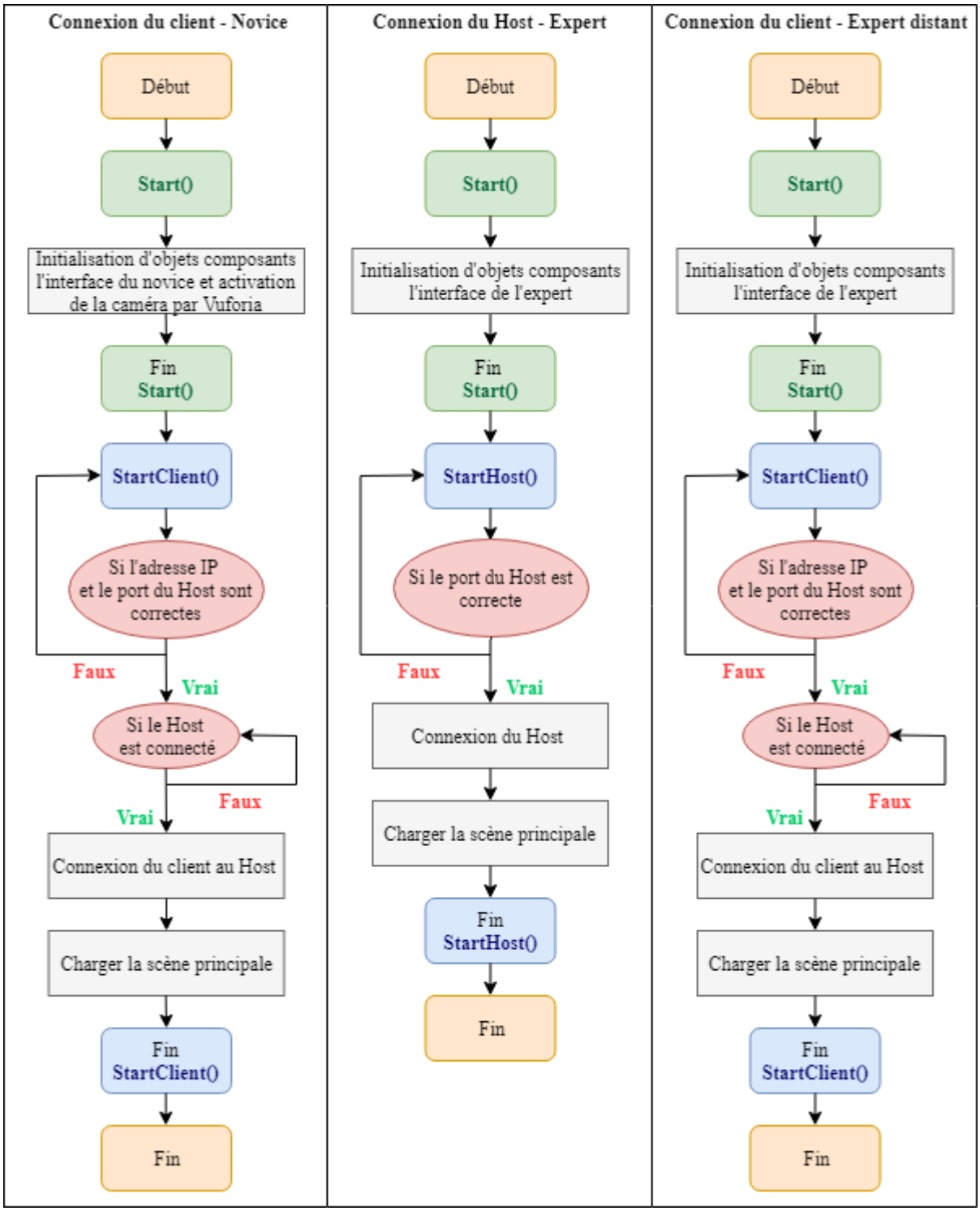


Figure 27 : Diagramme d'ouverture de connexion du système

3.5.3.6 Partage des identifiants des clients :

L'étape très importante afin que le transfert des données puisse se faire une fois les clients connectés au host, c'est l'étape d'association des identifiants réseaux '*NetworkIdentity*', nous la décrivons dans le schéma suivant :

En Local :

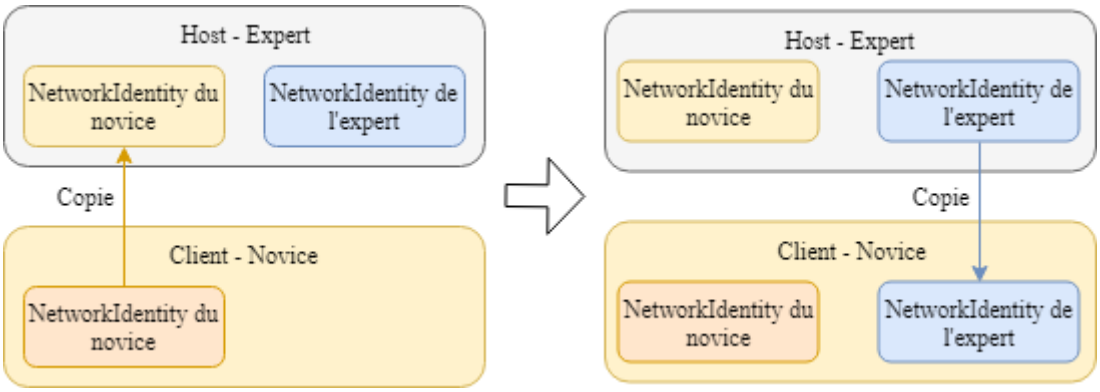


Figure 28 : Etapes de distribution des NetworkIdentity du client et le host

A distance :

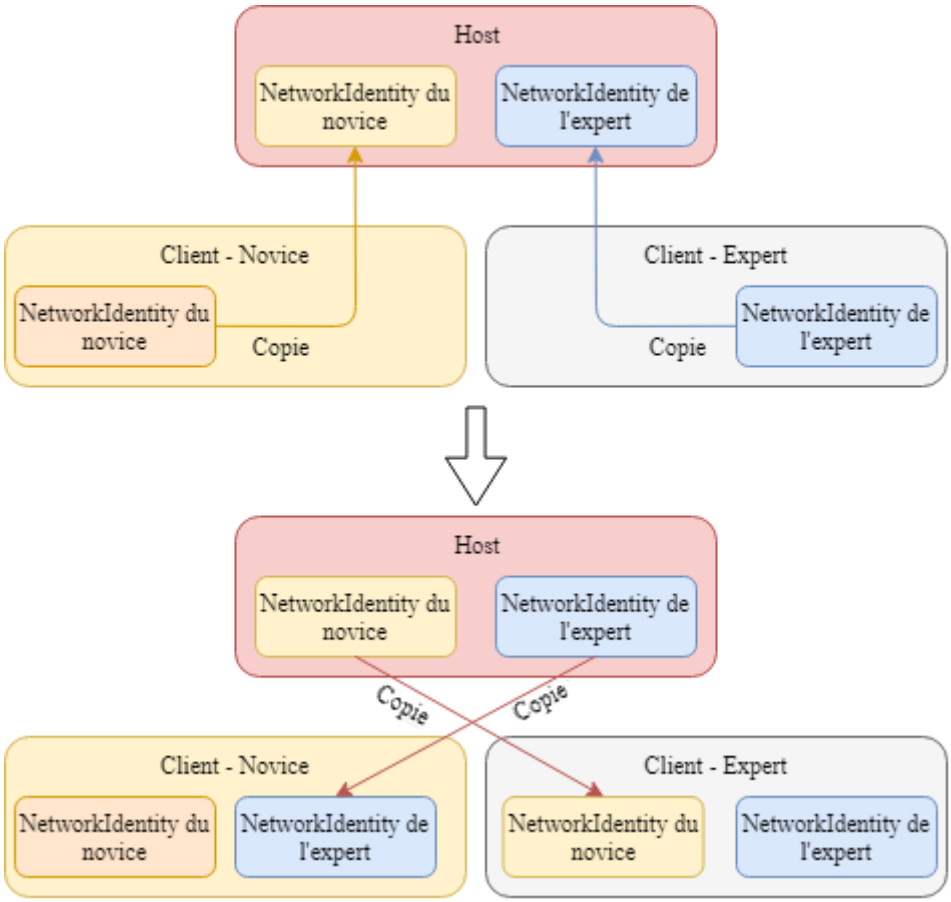


Figure 29 : Etapes de distribution des NetworkIdentity des clients établie par le host

3.5.3.7 Transfert des données sur le réseau :

En local :

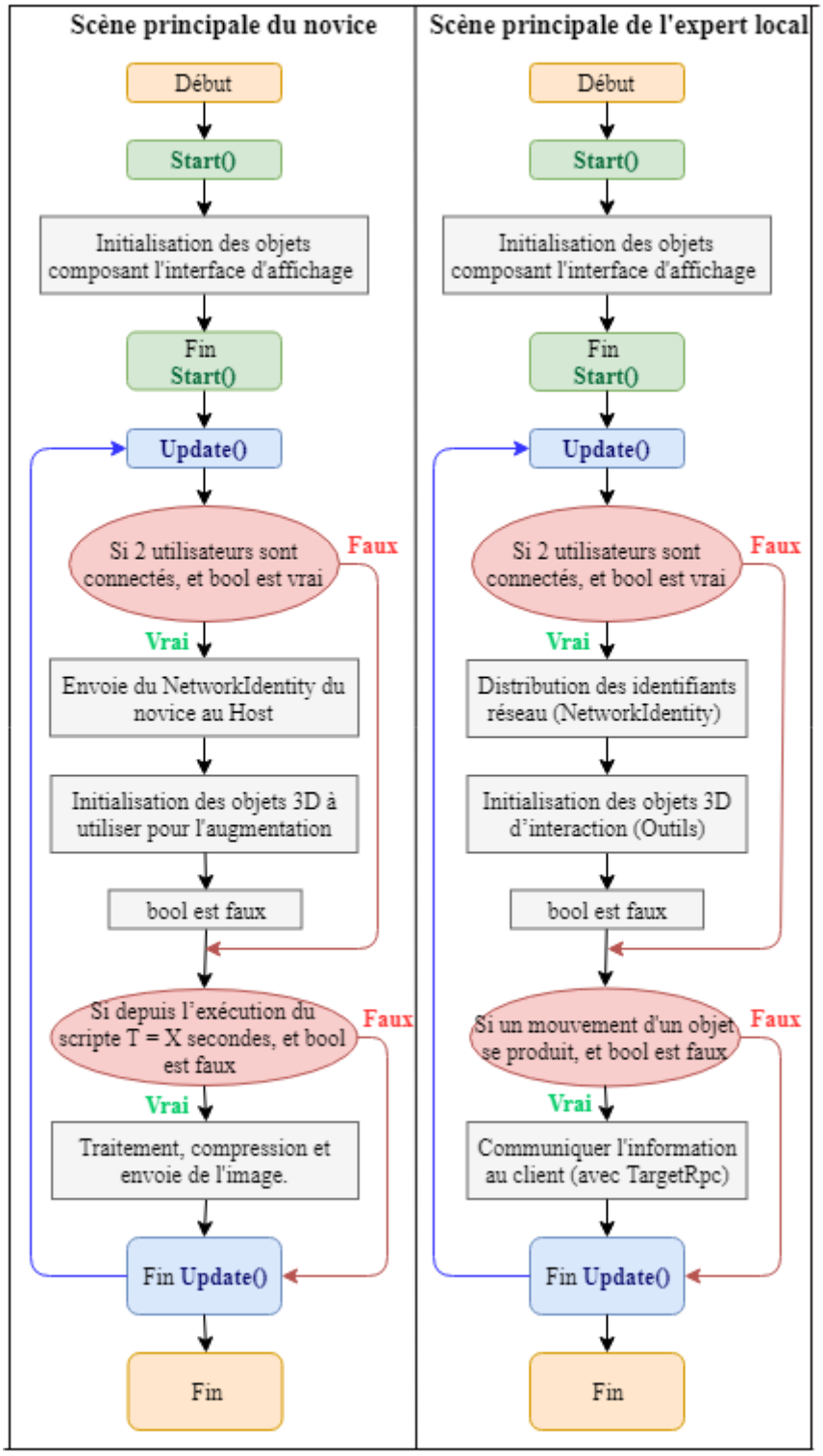


Figure 30 : Diagramme de traitement et d'envoi de données en local

A distance :

La scène du novice est la même que sur le diagramme précédent.

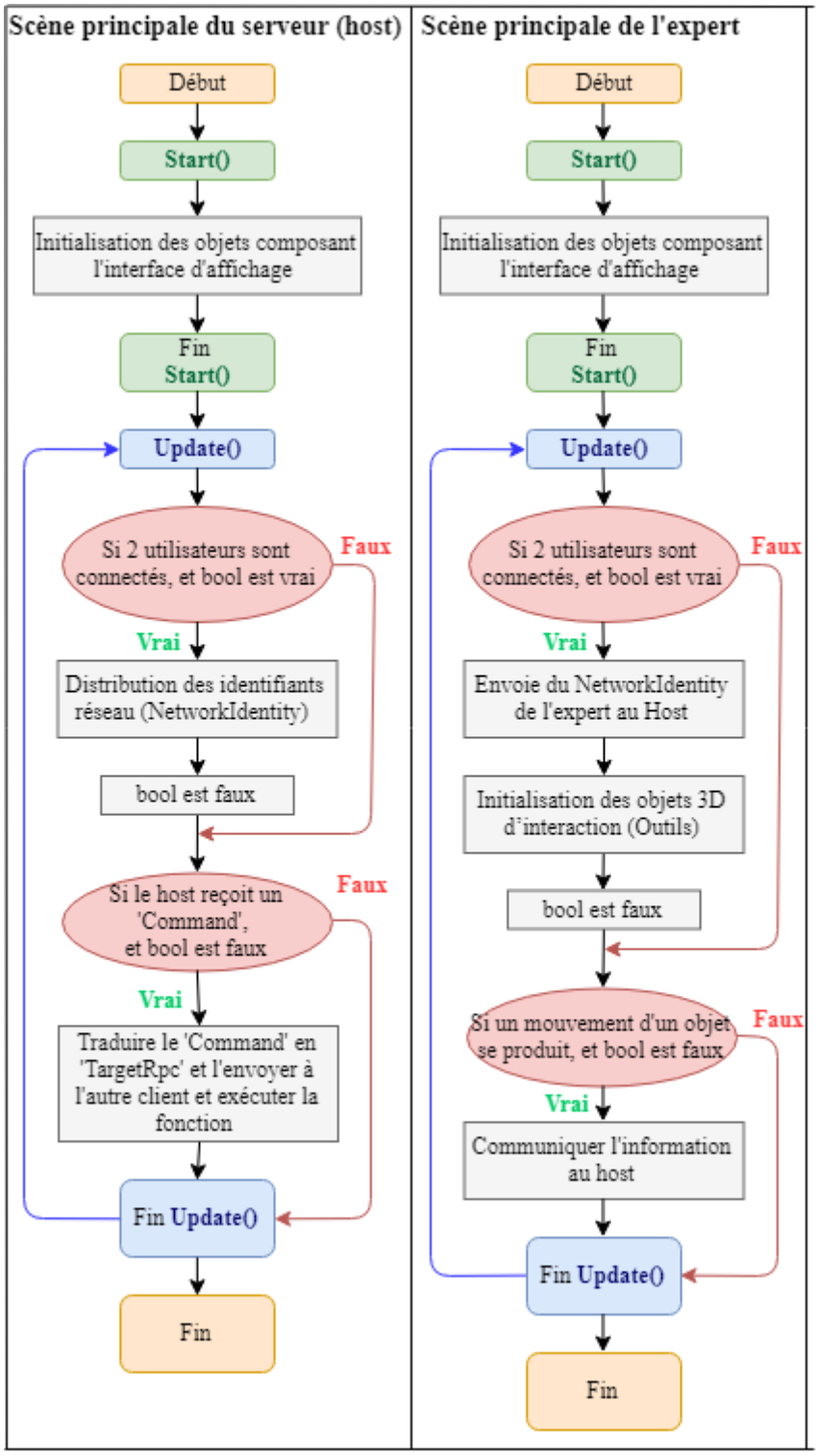


Figure 31 : Diagramme de traitement et d'envoi de données à distance

3.5.3.8 Avantages et inconvénients :

Nous estimons que c'est le système hybride réalisé avec Unet qui répond le plus aux attentes du CDTA car ce dernier permet d'installer le serveur au sein du CDTA, et que ce dernier ne

dépend pas d'internet, il peut être utilisé dans un cas où nous ne disposons pas de connexion internet.
L'inconvénient c'est que l'adresse IP du host et son port doivent être connus et doivent demeurer fixes pour que les clients puissent s'y connecter, afin de remédier à celle-ci, une solution comme Photon pourrait être utile pour transférer l'adresse IP et le Port pour plus de transparence vis-à-vis des utilisateurs.

Remarque : Nous avons donné cette appellation à ce système non pas car son architecture est hybride, nous faisons allusion au fait que le host joue deux rôles différents, dans chacune des situations : expert présent au CDTA, et expert distant.

3.6 Stockage et traitement des données de l'environnement augmenté :

Pour les augmentations nous utiliserons un SDK (software development kit) de réalité augmentée, il se chargera d'activer la caméra et devra détecter le marqueur, pour ensuite associer les objets virtuels à la scène réelle.
Tous les objets 3D qui composent l'environnement augmenté sont stockés en local dans le disque dur de la machine **du novice**, ceux-là vont juste être contrôlés par l'expert à distance, mais c'est en local que tous les traitements se passent, déplacement, translation, rotation, activation et désactivation des objets.
Ceci est justifié par le fait que ces données sont de taille importante et qui n'est pas intéressant de les envoyer via le réseau qui pourra vite finir par saturer la bande passante et cela ne fera qu'augmenter la latence/temps de réponse et l'effet du goulot d'étranglement.

3.7 Traitement d'images avant l'envoi avec Unity :

Etant donné que c'est le novice qui doit filmer la scène et envoyer les frames spontanément, une caméra intégrée à sa machine est nécessaire.
Afin d'envoyer des images filmées par la caméra, Unity permet d'extraire des textures '*WebCamTexture*' à partir de cette dernière '*WebCamDevice [ ]*', il faut donc :

- Activer la caméra avec l'option : *play ( )*
- Vérifier si un temps X secondes est passé depuis l'exécution du programme afin de contrôler le nombre de frames par secondes à traiter et envoyer (FPS), et ceci grâce à deux variables :

Temps = variable représentant le temps écoulé depuis l'exécution

FPS = variable représentant le nombre de frames à envoyer par seconde

Ainsi la fréquence d'envoi est contrôlée par le FPS, nous pouvons envoyer 60 FPS (consomme beaucoup la bande passante) comme nous pouvons nous permettre d'envoyer que 10 FPS car la e-maintenance n'exige ni la haute définition ni une fluidité importante comme dans les films d'action par exemple.

- Vérifier aussi si l'image dans le tampon vidéo récoltée par la caméra a changé depuis la dernière image ou pas avec la fonction suivante : ***didUpdateThisFrame***

  En vérifiant cette valeur avant d'effectuer tout traitement ça nous évitera d'effectuer un traitement vidéo coûteux plusieurs fois de plusieurs frames identiques, ceci nous permet de gagner dans la capacité de la machine car le traitement vidéo est gourmand en ressources GPU mais aussi à travers le réseau, ainsi si l'image n'a pas changé, il n'y a pas lieu de l'envoyer ainsi si l'utilisateur qui ne fait pas beaucoup de mouvement à la caméra aura une latence beaucoup plus réduite que quelqu'un qui utilise la caméra pour danser ou filmer des match de foot ou autre évènement où la scène change tout le temps, ou lors de son utilisation en réunion de travail par exemple, les membres pourront jouir d'une latence encore plus réduites car les seules mouvement qu'ils feront seront ceux de la présentation.

Jusqu'ici nous n'avons utiliser aucun SDK pour réalité augmentée, celui-ci va dominer la caméra, et ne permettra pas de l'utiliser car elle sera activée par lui durant toute la procédure, donc rien ne sert d'ajouter les fonctionnalités citées auparavant car elles sont prises en charge par le SDK, par contre nous garderons la fréquence d'envoie (FPS) pour que l'envoie des frames soit contrôlé.

Nous avons utilisé plusieurs SDK de réalité augmentée lors de nos réalisations, NyARToolKit, XZIMG et Vuforia et nous avons opté pour Vuforia pour des raisons de performance, de compatibilité et de rapport qualité/prix.

Pour envoyer les frames, nous sommes passés par plusieurs étapes assez importantes :

- Une fois la caméra activée par Vuforia, et que le marqueur est détecté dans la scène, les images sont capturées et traitées une à une et sont affichées sur une *texture 2D* sur le moteur Unity afin que les augmentations puissent être rajoutées (Objets 3D).
- Une caméra virtuelle est rajoutée afin de filmer la scène augmentée par Vuforia, c'est-à-dire : la scène réelle capturée contenant les objets 3D. Voici un schéma décrivant les étapes avant la compression :

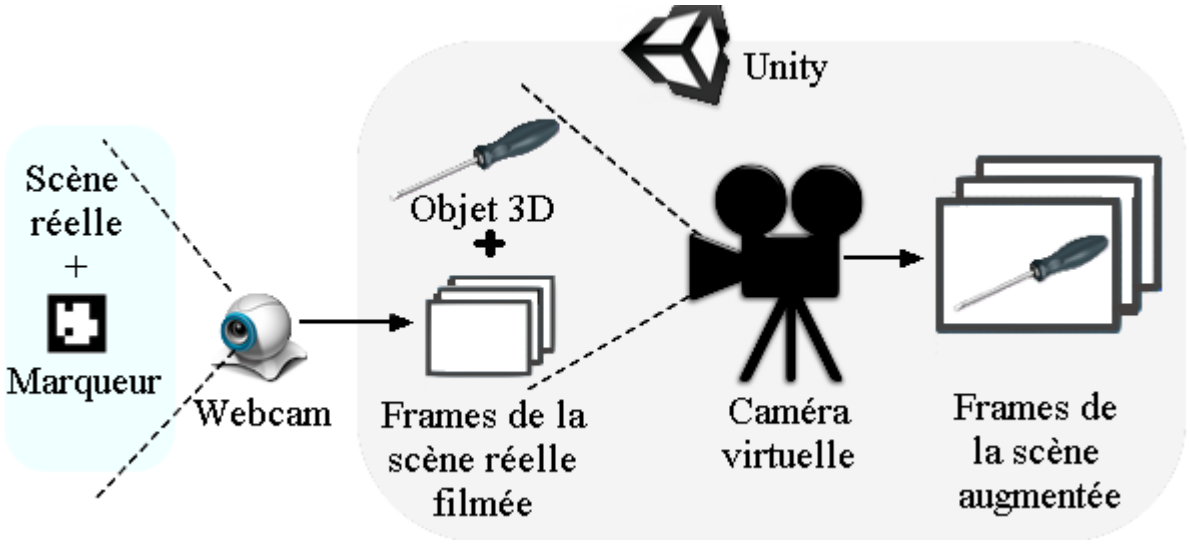


Figure 32 : Méthode utilisée pour extraire les images de la scène augmentée

- Une fois que nous avons les frames (images) de la scène augmentée, nous devons veiller à ce que chaque frame étant l'entité la plus lourde à transmettre, doit passer par une étape primordiale qui est souvent utilisée par les codecs vidéos, qui est la compression. Cette étape importante vise à compresser l'image avant d'être envoyée sans pour autant trop dégrader sa qualité, car certaines images font une taille supérieure à la normale de 200 ko (soit 200 ko par frame), certaines moins sans compression.

N'ayant pas la possibilité d'utiliser un codec, nous sommes obligés de convertir les frames une à une en format JPEG (ou JPG) et les compresser avant de les envoyer et ceci grâce à différentes méthodes de compression et décompression possibles, voici quelques fonctions importantes que nous utilisons :

- ***texture.ReadPixels();*** ➔ Pour lire les pixels de la texture à traiter
  texture.EncodeToJPG (TauxDeCompression)); ➔ Pour encoder les pixels de la texture en format JPG en spécifiant comme argument le taux de compression, le résultat retourné par la fonction est un tableau d'octets de type '*byte []*'.
- ***Destroy (textute);*** ➔ Apres l'envoie, on détruit la texture pour libérer l'espace.
- ***textureRecue.LoadImage();*** ➔ Utilisée à la réception pour décompresser les octets en fichier JPG.

Le taux de compression est variable ce qui permet de varier la qualité de l'image de 1ko/unité de temps jusqu'à 30ko/unité, ce qui semble être correcte pour le réseau.

En résumé quand le technicien filme la scène, nous y intégrons l'augmentation, nous filmons la scène augmentée (comme expliqué dans la figure **21**) grâce au moteur de jeu Unity3D, ensuite

nous traitons et compressons l'image en JPG (**Figure 22**) pour l'envoyer sous forme de bytes à l'expert.

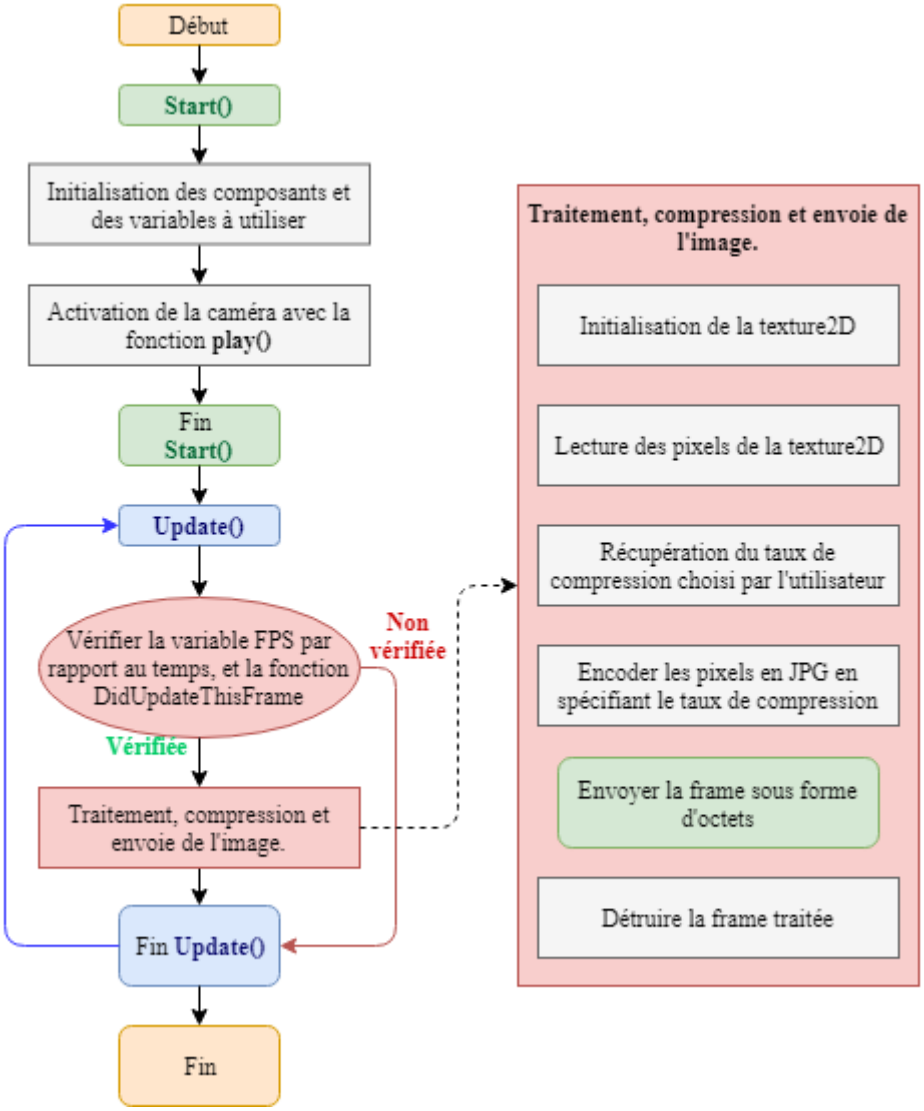


Figure 33 : Diagramme de traitement et compression de l'image

Cette conception nous permet d'éviter certaines contraintes liées à la réalité augmentée qui peuvent consommer beaucoup de ressources GPU et nécessite de la précision et du traitement d'image et la maitrise du langage HLSL (High Level Shader Language) qui est le langage de programmation des pipelines des cartes graphiques 3D intégré à l'API Direct3D, donc au lieu d'instancier l'objet chez les deux utilisateurs et synchroniser leurs positions par rapport à la scène réelle chez l'expert, c'est-à-dire que l'envoie vidéo et la synchronisation des objets s'envoient en même temps mais séparément, nous mettons donc l'objet 3D chez le technicien seulement pour éviter toute désynchronisation.

Voilà pourquoi l'application du novice est la plus gourmande en ressources GPU et de bande passante.

3.8 Contrôle d'objets 3D :

Le contrôle des objets 3D se fait par l'expert, son application de l'expert sera différente de celle du technicien, elle consomme beaucoup moins de ressources GPU, ainsi que moins d'espace, l'expert n'aura pas besoin d'une caméra, mais d'un écran pour lui permettre de visualiser la scène du technicien.

Il aura à sa disposition des outils d'interaction sur l'écran qui lui permettront de contrôler les objets 3D distants (présents sur la machine du technicien).

Afin de contrôler un objet 3D, l'expert n'aura qu'à mentionner avec le clavier du PC par exemple où il veut déplacer tel ou tel objet, et ceci en envoyant des appels RPC prédéfinies qui ne contiennent aucun argument, il y'aura donc certes beaucoup de fonctions, mais ça reste mieux que de les charger d'arguments.

Prenons pour exemple une méthode qui doit exécuter le déplacement d'un objet 3D vers la direction gauche, si l'utilisateur envoie comme argument de la fonction RPC la chaine de caractère ''Direction gauche'', une fois reçue chez l'utilisateur distant, la chaine de caractère est comparée avec d'autres pour enfin savoir quelle partie du code de la fonction va être exécuter.

Mais si le client appelle directement une fonction RPC qui se charge de faire le déplacement vers la gauche, celle-ci sera sans arguments, et donc quasi-vide.

Nous avons donc opté pour cette dernière méthode pour les appels RPC afin d'éviter d'encombrer la bande passante et de minimiser la latence.

3.9 La rétro-ingénierie :

La rétro ingénierie (reverse engineering) consiste à décomposer (décompiler) un logiciel pour avoir le code source, ainsi faire la fonction inverse du concepteur, car ce dernier a un code source et il le build pour obtenir un exécutable, l'ingénierie inverse consiste à obtenir ce même code à partir de l'exécutable (exe, apk etc.).

Lors du déploiement de l'application, une personne mal intentionnée pourrait être amené à récupérer le code source de notre application, pour contrer cela nous utiliserons l'obfuscation, Cette méthode rend le code source erroné pendant la décompilation. Il le remplace par des symboles, le code devient donc obfusqué mais toujours fonctionnel.

Ainsi lors de la décompilation, le code résultant devient tout simplement du charabia.

Le logiciel **Eazfuscator.NET nous** permet de protéger notre code source sans le casser, une fois installée il rend à chaque fois le code illisible mais fonctionnel, il permet aussi la

virtualisation du code, c'est-à-dire remplacer les instructions .NET d'un assembly par quelque chose d'inconnu à un observateur externe, mais équivalente au programme original pendant l'exécution.

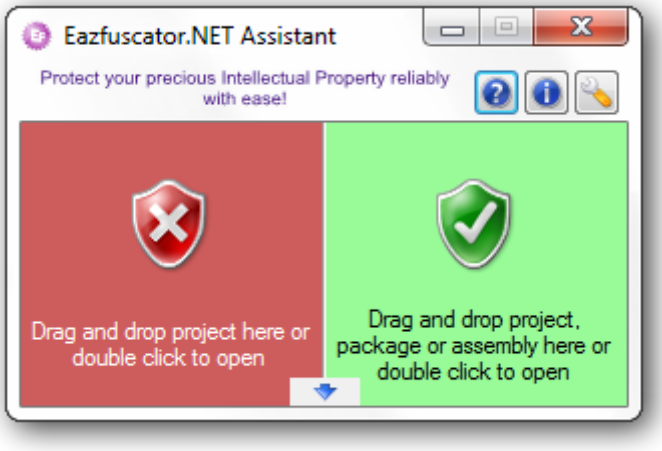


Figure 34 : Capture du logiciel Eazfuscator.NET

3.10 Plateformes :

- La plateforme d'envoi, celle du technicien novice est principalement sur mobile ou tablette disponible en version Android et IOS.

- La plateforme de réception, celle de l'expert est sur PC (Windows et Mac OS et Linux) afin que l'expert puisse guider et gérer en ayant tout ce dont il a besoin.

3.11 Conclusion :

Après avoir conçu plusieurs architectures réseaux différentes, et tester différents protocoles, ainsi que plusieurs façons de partage vidéo, nous avons opté au final pour le système hybride qui semblait correspondre le plus aux attentes du CDTA, et nous avons décidé d'utiliser que les outils proposés par Unity afin de tester ses limites.

Pour envoyer un flux vidéo (streaming), plusieurs applications utilisent des codecs vidéo afin de faciliter l'étape de compression et décompression, malheureusement Unity étant un puissant moteur de jeux 3D, et d'applications de réalité virtuelle et augmentée, est limité car ce dernier ne permet pas de réaliser une architecture de type pair à pair ou d'utiliser des protocoles réseaux tel que le RTP qui est l'idéal pour un transfert vidéo en temps réel.

En effet, il est possible d'utiliser *« VideoPlayer »* de Unity qui utilise les codecs-vidéo H.264 et VP8, mais le composant « *Video Source »* de ce dernier ne permet que la lecture d'un fichier vidéo stockée en locale, c'est-à-dire qui est déjà enregistrée sur la machine, ou alors à partir d'un lien URL comme par exemple d'une vidéo *Youtube*.

Les deux méthodes semblent donner des résultats trop lents pour une application temps réel.

Pour conclure, le choix de séparer les deux applications (celle du novice et celle de l'expert), est due à la lourdeur de l'application du novice (car c'est elle qui se charge des traitements d'objets de la réalité augmentée ainsi que l'envoie des frames) et elle fait environs 95 Mo, alors que l'application de l'expert est d'une taille de 10 Mo. Nous avons donc jugé que les séparer était la meilleure chose à faire.

Chapitre 4

4 Implémentation d'une application de e-maintenance

4.1 Introduction :

Dans le chapitre précédent, nous avions vu tous les différents systèmes que nous avions réalisé, bien évidement il fallait que nous fassions un choix, notre choix se base sur plusieurs critères, d'un côté nous devons avoir un serveur fixe au sein du CDTA, afin qu'on puisse l'administrer à leur guise, et d'un autre côté, l'idéal pour transférer les frames d'une entité à une autre c'est une architecture pair à pair, Unity, étant limité, ne permet pas d'initier la connexion des clients à travers le serveurs puis laisser le transfert d'images se faire d'un client à l'autre, nous avons choisi le système hybride réalisé avec Unet, car c'est celui qui convient le plus à l'organisme du CDTA, et que nous allons détailler dans ce dernier chapitre que nous finirons par des résultats de tests sur l'application.

4.2 Outils de développement :

4.2.1 Unity 3D :

Unity 3D, réalisé par Unity Technologies, est une suite complète d'outils de développement de jeux vidéo et tout autre type de contenu interactif 3D et 2D. Le logiciel dispose d'une interface d'intégration intuitive pour importer des modèles 3D, des textures, des sons etc.

En plus de gérer la logique d'une application, l'affichage graphique (2D ou 3D), l'audio, les collisions des objets, le moteur de jeu possède aussi un éditeur de scène qui permet de visualiser des portions de l'application sans avoir à compiler.

Bien que non libre, la version gratuite du logiciel est amplement suffisante pour réaliser quelque chose d'intéressant en permettant l'accès à déférents périphériques des dispositifs mobiles (caméra, microphone, capteurs, etc) et proposant de nombreuses bibliothèques (gratuites et payantes), des animations, des personnages, des décors, etc. Il est aussi possible d'intégrer des ressources comme les plugins de réalité augmentée comme ARToolkit, Vuforia…, et aussi des plugins réseau comme Unet ou Photon.

Basé actuellement sur le langage de programmation C#, Unity permet de facilement écrire des scripts afin de gérer l'interaction avec l'utilisateur, les déplacements des objets etc. Il intègre un éditeur de scripts, MonoDevelop permettant de développer en C# mais aussi en JavaScript.

L'un des points forts et le plus important de Unity c'est qu'il propose la possibilité d'exporter gratuitement tout contenu développé vers de nombreuses plateformes sans avoir à modifier le code source, voici quelques plateformes supportées par Unity :

- Windows/Mac/Linux
- iOS/Android/Windows Phone
- Playstation 3/4/Vita
- Xbox 360/Xbox One
- Wii/Wii U
- Navigateurs internet

4.2.2 Langage C# :

Le langage C# (C sharp) est un langage de programmation orienté objets fortement typé et créé par Microsoft en 2002. Il est dérivé du C++ et sa syntaxe ressemble un peu à celle du Java, ce sont d'autres langages de programmation très populaires.

C# est le langage de programmation le plus utilisé de Microsoft en 2015. Il n'est pas multi-plateformes, compile et s'exécute uniquement sur les systèmes d'exploitations Windows. Cependant, grâce au fameux logiciel Unity et son complément MonoDevelop, il permet d'écrire des codes sources en C#, sur Windows ou sur Mac OS X. Et Unity 3D permet d'exporter les projets sur tous types d'appareils PC et mobiles, ça fonctionnera sans problème n'importe où.

4.2.3 MonoDevelop :

MonoDevelop est un environnement de développement libre de GNOME conçu pour le langage C# et d'autre langages .Net. Il est fourni avec le logiciel Unity, il sert à combiner le fonctionnement familier d'un éditeur de texte avec des fonctionnalités supplémentaires pour le débogage et d'autres tâches de gestion de projet.

4.2.4 Unet :

Unet est un moteur de réseau développé par Unity Technologies, il est présenté sous forme d'une bibliothèque réalisée en C++ et C#.

Le système Unet est conçu pour fonctionner qu'avec le logiciel Unity, il permet de construire des infrastructures réseau en se basant sur des architectures Client-Serveur ou Client-Host, le réseau peut fonctionner en hors ligne et en ligne.

L'avantage avec Unity Networking c'est qu'il est rapide à déployer et hautement personnalisable, et qu'il permet de tester rapidement des fonctionnalités sur le réseau.

4.2.5 Photon (PUN) :

Photon est un moteur de réseau indépendant, le SDK (Software Development Kit) de Photon est développé par la société allemande Exit Game, garantissant rapidité (faible latence), fiabilité, et scalabilité.

PUN (Photon Unity Networking) est le SDK destiné spécialement à Unity, il propose un service entièrement géré par des serveurs locaux, s'exécutant dans des régions du monde entier à tout moment.

Il dispose d'une interface de développement conviviale et adaptée aux besoins des utilisateurs, basée sur le principe de Room, il s'agit de faire correspondre les utilisateurs à une session de partagée appelée 'Room', afin qu'ils puissent partager des messages de manière synchrone, en temps réel. Dans chaque Room il y a un master client par room, généralement c'est le premier qui se connecte et crée la Room, mais ce dernier va être remplacé par un autre utilisateur s'il se déconnecte du réseau ou de la Room, le partage peut se faire via divers protocoles de communication à savoir UDP, TCP, HTTP, WebSocket, et RPC, et ceci sur toutes les plates-formes. Tous les SDK clients peuvent interagir entre eux, peu importe si leur machine est un Windows, iOS, Android, Web, console ...etc.

4.2.6 NyARToolkit :

Le projet NyARToolkit développe plusieurs bibliothèques de réalité augmentée basées sur la vraie version d'ARToolKit. Les bibliothèques s'exécutent dans différents environnements gérés par le langage Java et le C# principalement et aussi les environnements de développement comme Unity et Processing.

Les bibliothèques NyARToolkit actuelles ont les API ARToolKit Professional (ARToolKit5).

NyARToolkit for Unity est une bibliothèque NyARToolkit qui s'exécute sur la plate-forme Unity. Il est purement en langage C#, il fonctionne dans tous les environnements Unity, y compris WebPlayer [19].

4.2.7 Vuforia :

Vuforia est une plate-forme logicielle pour créer des applications de réalité augmentée. Unity intègre le SDK (un kit de développement de logiciels) de Vuforia, qui permet de faciliter la création d'expériences de réalité augmentée à la pointe de la technologie, les développeurs peuvent facilement ajouter des fonctionnalités avancées de vision par ordinateur à n'importe quelle application, ce qui leur permet de reconnaître des images et des objets (marqueurs), et d'interagir avec des espaces dans le monde réel.

Vuforia est un produit payant ayant des fonctionnalités diverses, mais il existe une version gratuite afin de tester ses performances.
La plate-forme Vuforia prend en charge le développement d'applications AR dans plusieurs plateformes, voici la liste des versions supportées par Vuforia [20] :

- Android 4.4+
- iOS 9+
- Windows 10 uniquement (Réalisable sur Visual Studio seulement)

4.2.8 Augmented Vision de XZIMG :

XZIMG Augmented Vision est un plugin multi-plateforme pour réalité augmentée conçu pour être utilisé par Unity similaire à ARToolKit l'original, mais non libre, il est sous forme d'une bibliothèque qui offre une détection d'objets à l'aide d'un flux vidéo d'entrée. Il produit une position 3D précise et l'orientation (pose) des objets détectés.
Dans la version d'essai de XZIMG Augmented Vision Solution, la détection est intentionnellement interrompue de temps en temps.
XZIMG Augmented Vision, peut détecter des marqueurs noires et blancs ayant des formes, et des images naturelles.

Figure 35 : Types de marqueurs détectables par XZIMG Augmented Vision [18]

4.2.9 Visual Studio :

Visual Studio est un environnement de développement intégré (IDE) complet pour Android, iOS, Windows, Internet et le cloud.

Il permet d'écrire de coder avec précision et efficacité, sans perdre le contexte du fichier en cours. Il permet aussi de zoomer sur des détails tels qu'une structure d'appel, des fonctions associées, des archivages et un état de test, il possède un débogueur pour détecter et corriger rapidement des bogues dans divers langages afin de détecter et diagnostiquer des problèmes de performances, sans quitter le flux de travail de débogage.

4.3 Description du système :

Notre système de e-maintenance est réalisé pour aider des apprentis dans des domaines techniques dans une grande entreprise par exemple, le technicien doit être guidé par un expert qui ne peut pas se déplacer de son bureau ou dans un cas où il doit intervenir dans deux opérations de maintenance différente en même temps.

Le technicien doit disposer d'un smartphone ou tablette avec caméra intégrée car l'application va filmer la scène, et d'un marqueur pour charger les modèles 3D elle doit aussi permettre de recevoir des indications de la part de l'expert qui doivent apparaître sur l'écran afin de savoir comment et quelle partie de l'engin il faut réparer, une fois terminé avec un outil il doit l'indiquer à l'expert grâce à un bouton 'suivant'.

L'expert aura besoin d'un ordinateur en général afin de bien visualiser la scène, un clavier et une souris pour plus de précision en contrôlant les objets 3D pour guider le technicien, il pourra choisir l'outils à utiliser, le déplacer ainsi que le faire tourner.

Les utilisateurs peuvent se connecter dans un même réseau local en se connectant au réseau de l'entreprise par exemple, ou alors à internet si l'expert est distant.

4.4 Utilisation de l'application :

4.4.1 Rôles des utilisateurs :

Nous avons deux acteurs principaux, l'expert et le technicien ayant deux rôles différents, et cela dans deux cas possibles, le 1er est le cas local, quand l'expert est présent dans l'entreprise, le 2ème est quand l'expert est distant :

Cas local	Cas distant
L'expert : Doit se connecter à l'application locale comme **host** en introduisant un numéro de port libre et attendre que le technicien se connecte afin de commencer l'opération.	**L'administrateur :** Remplace le post de l'expert local pour se connecter sur l'application en tant que **host intermédiaire** seulement, il introduit un numéro de port libre et attende que les clients (expert distant et technicien) rejoignent sa session.
Le technicien : Se connecte à son application **cliente** en entrant le port et l'adresse **IP locale** du host**,** et ceci après la connexion de ce dernier.	**L'expert distant :** Se connecte à son application **cliente** qu'une fois le host connecté en introduisant son port et son adresse **IP publique**, et ne peut communiquer aucune information tant que le technicien ne s'est pas encore connecté.
	Le technicien : Se connecte à son application **cliente** de la même manière que dans le cas local.

Tableau 2 : Connexion au système chez les différents utilisateurs

Une fois les utilisateurs connectés, l'opération de e-maintenance peut commencer :

4.4.1.1 Le technicien :

- Filme la machine à réparer.
- Met en évidence le marqueur (l'image) afin qu'il soit facilement détectable par Vuforia
- L'envoie des images capturées se fait automatiquement.
- S'il reçoit une indication de la part de l'expert, il peut appuyer sur le bouton 'suivant' une fois qu'il a terminé d'exécuter l'action demandée.
- Il peut aussi régler le taux de compression et le nombre de frames à envoyer par secondes au dépend de la bande passante.

4.4.1.2 L'expert :

- Visualise la scène filmée par le technicien pour analyser le problème.
- Choisi un outil ou une indication de sa barre d'outils (avec sa souris ou son clavier) et sa position.
- En choisissant l'outil, il s'activera chez le technicien.
- Il peut lui changer de place, le faire tourner, ou le remplacer à sa guise pour guider le technicien.
- Quand le technicien appuie sur 'suivant', il peut passer à la suite de l'opération et donc choisir un autre outil etc…

4.4.2 Modèles 3D et marqueur :

Afin que l'expert puisse guider le technicien, nous devons posséder des outils 3D, ils peuvent être créé par un modélisateur ou un infographe, ou alors être téléchargé à partir de l'Asset Store de Unity, nous pouvons trouver des modèles gratuits, mais les meilleurs sont payants.
Nous avons utilisé certains gratuits comme les outils suivants :

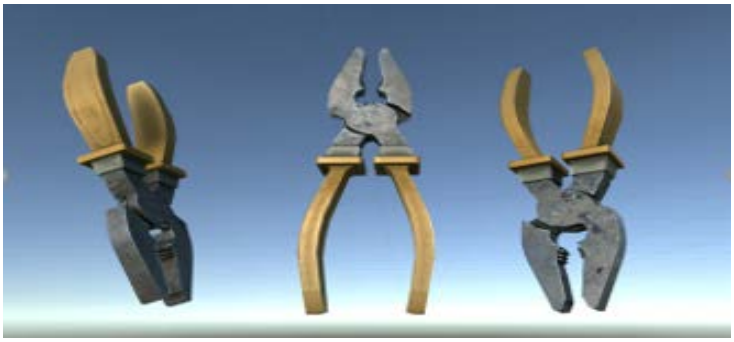


Figure 36 : Modèle 3D d'une pince

Figure 37 : Modèle 3D d'une clé à griffe

Figure 38 : Modèle 3D d'un marteau

Il existe aussi plusieurs marqueurs images prédéfinis offerts par Vuforia que nous pouvons utiliser, comme il est possible aussi d'en personnaliser un, voici le marqueur que nous utiliserons :

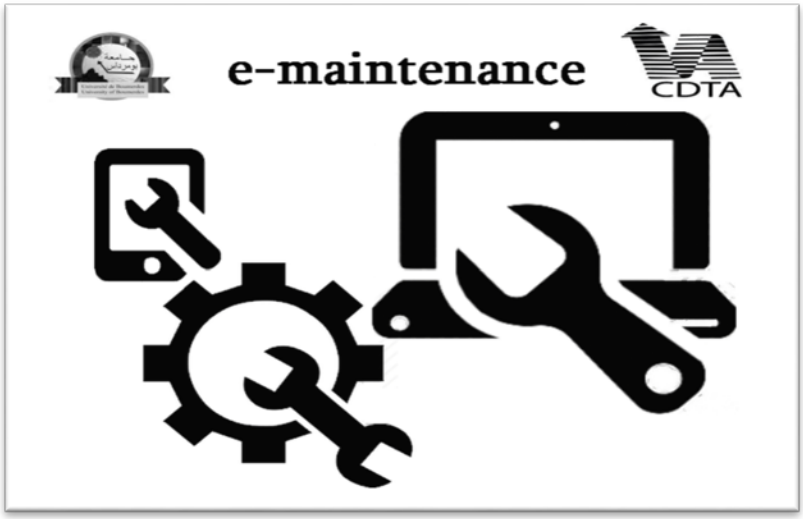


Figure 39 : Marqueur utilisé pour notre application

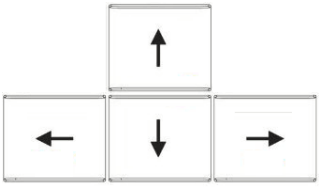
4.4.3 Interaction :

Souris : Le clic droit de la souris de l'ordinateur sert à :

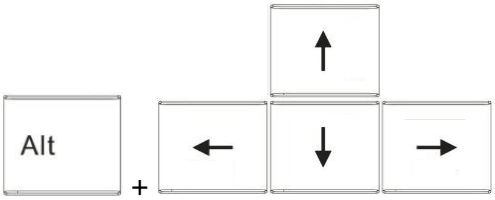
- Cliquer sur les champs à remplir
- Sélectionner l'outil à choisir
- Cliquer sur les boutons de l'application

Touches du clavier :

- Pour déplacer l'objet :



- Pour faire tourner l'objet :



Ecran tactile d'un smartphone ou une tablette : Le technicien utilise son doigt pour saisir les champs à remplir et appuyer sur les boutons de l'application

4.4.4 Scénario d'utilisation :

4.4.4.1 Connexion des utilisateurs :

a) Expert :

Figure 40 : Interface de connexion de l'expert

L'expert se connecte sur un ordinateur en introduisant un numéro de port libre : 

et clique sur le bouton connexion : 

Il entre dans la scène principale, l'écran est blanc car le technicien ne s'est pas encore connecté :

Figure 41 : Interface principale de l'expert avant connexion du technicien

b) Technicien :

Figure 42 : Interface de connexion du technicien

Le technicien se connecte avec son smartphone en entrant l'adresse **IP locale** 

et le port 

de l'expert, et clique sur connexion 

.

4.4.4.2 Partage d'images de la scène :

Voici ce que nous pouvons apercevoir quand on se connecte en tant que technicien, la scène filmée par la caméra, nous pouvons apercevoir que la même scène filmée est affichée sur l'interface de réception (celle du technicien) :

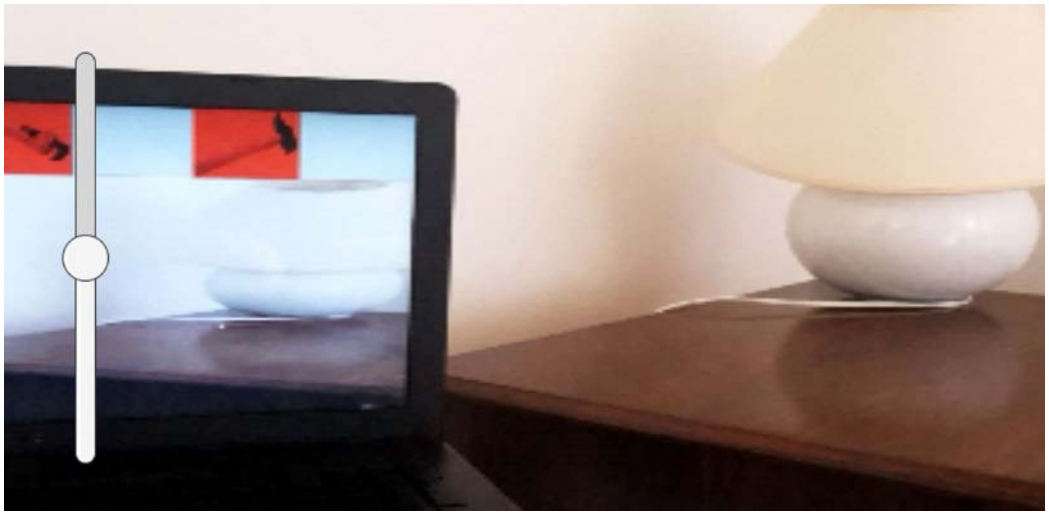


Figure 43 : Interface principale du technicien

4.4.4.3 Variation de la qualités d'image :

Nous pouvons voir les différentes qualités de l'image envoyée :

Figure 44 : Qualité d'image pas très lisse mais acceptable

Figure 45 : Qualité d'image moyennement lisse

Figure 46 : Qualité d'image bonne

Remarques :

- Quand on réduit la qualité de l'image envoyée au maximum, l'image à la réception est médiocre, il devient donc difficile, voire impossible de visionner la scène.

Voici le résultat :

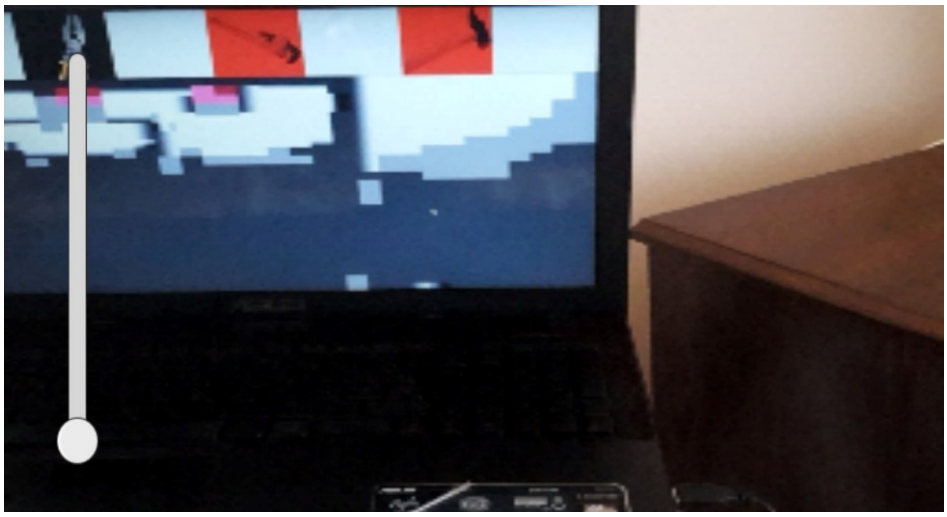


Figure 47 : Qualité d'image déformée

- Quand on met la meilleure qualité d'image, cela veut dire qu'on ne compresse presque pas l'image, est donc la taille envoyée est très importante et crée un petit retard à la réception.

4.4.4.4 Utilisation du marqueur :

Quand nous mettons notre marqueur sur les champs de vision de la caméra, un outil apparait :

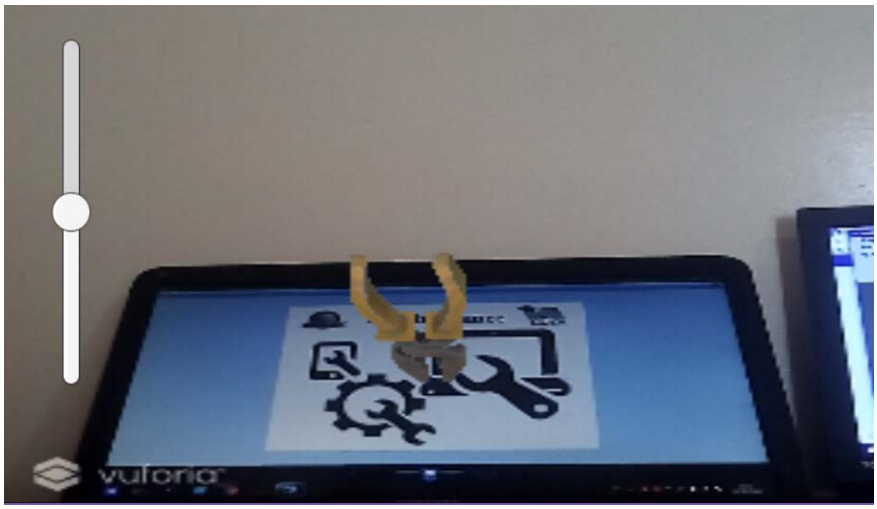


Figure 48 : Apparition d'un outil 3D à la présence du marqueur

4.4.4.5 Contrôle d'objets 3D :

Les objets 3D du technicien sont contrôlés par l'expert avec son clavier.

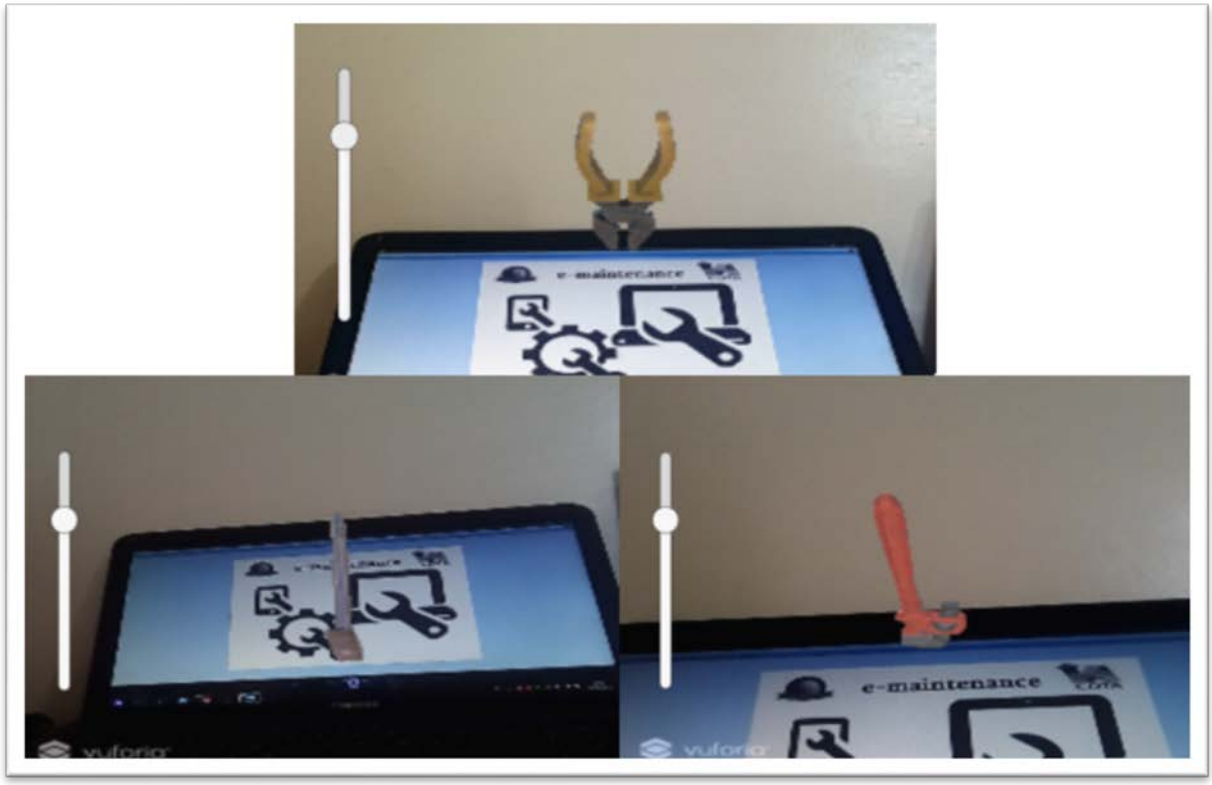


Figure 49 : Choix des outils

Il peut les déplacer afin d'indiquer au technicien où il faut mettre l'objet :

Figure 50 : Outil contrôlé par l'expert

4.5 Conclusion :

Dans le domaine industriel, il est indispensable d'exécuter des opérations de maintenance, qu'elles soient correctives ou préventives.

Le technicien chargé de réparer la défaillance n'est pas en mesure de maitriser la tâche dans certains cas, pour des raisons de complexité du matériel sur lequel s'effectue la maintenance, ou sur des équipements récents sur lesquels le personnel (technicien) n'a pas une formation approfondie, ou parfois ce manque de maitrise se traduit simplement par le manque d'expérience

du technicien, il nécessite donc l'assistance d'un expert afin qu'il puisse le guider en économisant le temps de déplacement car l'expert n'a pas à se déplacer de son bureau.

L'application que nous avons réalisée permet à l'expert de guider son novice sans se déplacer sur le lieu de l'opération, offrant une multitude d'outils gérés par l'expert qui lui permettent d'orienter et guider le technicien vers l'utilisation des bons outils et comment les positionner dans le bon sens et au bon endroit, ainsi le technicien, en filmant sa scène, reçoit instantanément les indications envoyées par l'expert qui visualise en temps réel tout ce que fait le technicien, notre système permet donc à l'expert d'économiser le temps de déplacement, de guider plusieurs techniciens, et offre au novice un cours pratique qui lui permettra d'acquérir des connaissances en ayant un bon résultat à la fin de son opération de maintenance.

En plus notre système est indépendant du réseau internet dans le cas où les utilisateurs sont tous connectés à un même réseau local, ils peuvent communiquer à travers ce dernier jouissant ainsi d'une latence très réduite.

Une communication à travers internet est possible, elle est utilisée dans le cas où l'expert est distant, le système permet d'offrir une synchronisation des commandes envoyées de l'expert vers le novice, et la scène est acheminée de manière fluide du novice vers l'expert en respectant la contrainte temps réel.

Offrant ainsi une solution optimale qui réunit gain de temps, qualité de résultat et simplification des tâches pour les deux entités : expert et technicien novice.

Conclusion générale et perspectives

Le développement des technologies a permis de faire évoluer l'industrie en augmentant son taux de productivité à un rythme très rapide, mais ce remarquable progrès a engendré des problématiques, car cela nécessite le bon fonctionnement du matériel industriel, et cela par des vérifications, des contrôles, et des réparations des pannes sans pour autant gaspiller du temps ou pénaliser d'autres équipements dépendant de ce matériel.

Nous avons proposé dans ce rapport une solution à cette problématique en combinant deux concepts : la e-maintenance et la réalité augmentée, pour permettre aux techniciens dans n'importe quel domaine pratique de se faire assister et guider par un expert peu importe la distance qui les sépare, leur permettant ainsi de gagner du temps, car l'expert peut tarder à arriver sur les lieux, et parfois, les techniciens ne peuvent se permettre de faire durer une panne trop longtemps, ou d'exécuter une réparation sans la présence d'un expert.

Nous avons réalisé un système de e-maintenance basé sur un environnement augmenté collaboratif, et un partage vidéo temps réel, offrant un ensemble d'outils d'interaction pour aider l'expert à guider le technicien sur le matériel à réparer, et cela en diffusant sa scène à l'expert avec un contrôle intégral de la qualité de service, pour assurer une bonne transmission, le processus de cette dernière est composé de plusieurs étapes, une compression d'image avant l'envoi et une décompression à l'arrivée, avec une possibilité de varier la qualité et le nombre de frames à traiter et à envoyer par seconde.

Notre système garantit aussi une bonne synchronisation et une latence réduite en envoyant des paquets légers, contenants des informations de petites tailles afin de ne pas encombrer la bande passante, de cette manière le système offre la possibilité d'envoyer des informations consécutives et instantanées sans engendrer des retards de transmission.

En plus d'assurer la synchronisation des objets 3D et le fonctionnement en temps réel, notre conception permet aussi de supprimer les contraintes liées à la distance, car notre système permet de rassembler des utilisateurs d'un même réseau local et d'autres utilisateurs distants se connectant via internet.

Dans les versions futures, nous ajouterons un système de gestion de plusieurs utilisateurs simultanés afin de permettre à l'expert de gérer et guider deux techniciens en même temps par exemple, et nous intégrerons un chat vocal, afin de simplifier encore plus la tâche au technicien, ces versions-là pourront apporter une bonne solution dans un temps très réduit.

Bibliographie

[1] https://jeanperrot.eu/en/innovation/augmented-reality-in-the-industry-smart-glasses, consulté en Mars 2018

[2] ANNE SEGUY, « DÉCISION COLLABORATIVE DANS LES SYSTÈMES DISTRIBUÉS -APPLICATION À LA E-MAINTENANCE », 5 décembre 2008

[3] Devices world, iSCADA:Implementing eMaintenance

[4] Levrat E., Salzemann B., Clanché F., Bron J.Y., "TELMA Plate-forme d'intégration de télémaintenance pour l'enseignement et la recherche", Journal sur l'enseignement des sciences et technologies de l'information et des systèmes (J3eA), 5 (HS 2), 2006

[5] https://cordis.europa.eu/result/rcn/87875_fr.html , consulté en Mars 2018

[6] Brigitte Chebel-Morello; Jean-Marc Nicod; Christophe Varnier, From Prognostics and Health Systems Management to Predictive Maintenance 2, Volume 7, 2017

[7] https://www.canon.fr/for_work/solutions/solutions/office_software/emaintenance/ consulté en Mars 2018

[8] Azuma et al., « Tracking in Unprepared Environments » Pour système de réalité augmentée, "Computers and Graphics, Vol. 23, no. 6, décembre 1999

[9] Vincent HAVARD ; M'hammed SAHNOUN ; Navonil MUSTAFEE ; Anna WIENKE ; Dorian BOULC'H ; Phil GODSIFF ; Andi SMART ; David BAUDRY, La e-maintenance et la réalité augmentée, 20 avril 2015

[10] GUEMOUGUI Abdessattar, Réalité Augmentée pour les Jeux Vidéo et les Serious Games, 26 mai 2013

[11] Lucie Masson, Suivi temps-réel d'objets 3D pour la réalité augmentée

[12] Christophe DEHAIS, Contributions pour les applications de réalité augmentée. - Suivi visuel et recalage 2D. - Suivi d'objets 3D représentés par des modèles par points, 21 mai 2008

[13] Nadia Zenati – Henda, Contribution à la conception et à la réalisation d'un système de réalité augmentée pour la maintenance, 30 juin 2008

[14] Timothy BURK, ENS de Lyon, cours - Techniques de diffusion vidéo sur l'Internet.

[15] Fadi Boulos, Transmission d'images et de vidéos sur réseaux à pertes de paquets : mécanismes de protection et optimisation de la qualité perçue, 16 avril 2010

[16] Nicolas MENECEUR, Quelques mots sur la technologie de streaming

[17] http://www.virtu-desk.fr/blog/divers/protocole-rdp-10-windows-server-2016-les-ameliorations-graphiques.html, consulté en mars 2018

[18] https://www.xzimg.com/Docs, consulté en juin 2018

[19] https://nyatla.jp/nyartoolkit/wp/ , consulté en juin 2018

[20] https://library.vuforia.com/articles/Solution/Vuforia-Supported-Versions.html, consulté en juin 2018

[21] RIAHLA Mohamed Amine, Cours Cryptographie et sécurité informatique - Chiffrement par bloc AES 2018

[22] https://doc.photonengine.com/en-us/onpremise/v3/reference/encryption, consulté en mai 2018

[23] http://sebsauvage.net/comprendre/encryptage/crypto_dh.html, consulté en mai 2018